

CENTRO UNIVERSITÁRIO CARIOCA - UNICARIOCA
CURSO DE BACHARELADO EM CIÊNCIA DA COMPUTAÇÃO

ISMAEL WESLEY NEVES DE BRITO

INTRODUÇÃO A LÓGICA DE PROGRAMAÇÃO POR MEIO DE
JOGOS INFANTIS

RIO DE JANEIRO

2021

B849i

Brito, Ismael Wesley Neves de.

Introdução a lógica de programação por meio de jogos infantis/
Ismael Wesley Neves de Brito. Rio de Janeiro, 2021.
50 f.

Orientador: Manuel Martins Filho.

Trabalho de Conclusão de Curso (Graduação em
Ciência da Computação) – Centro Universitário
Unicarioca. Rio de Janeiro, 2021.

1. Jogos educativos – Tecnologia. 2. Jogos infantis –
Tecnologia - Educação. I. Martins Filho, Manuel. II. Título.

CDD 371.397

ISMAEL WESLEY NEVES DE BRITO

**INTRODUÇÃO A LÓGICA DE PROGRAMAÇÃO POR MEIO DE JOGOS
INFANTIS**

Trabalho de Conclusão de Curso apresentado à Coordenação do Curso de Ciência da Computação do Centro Universitário Carioca como requisito parcial para obtenção do título de Bacharel em Ciência da Computação, sob orientação do professor Manuel Martins Filho.

RIO DE JANEIRO

2021

ISMAEL WESLLEY NEVES DE BRITO

**INTRODUÇÃO A LÓGICA DE PROGRAMAÇÃO POR MEIO DE JOGOS
INFANTIS**

Trabalho de conclusão de curso submetido à Banca Examinadora designada pelo Curso de Graduação em Ciência da Computação do Centro Universitário Carioca como requisito para obtenção do grau de Bacharel em Ciência da Computação.

BANCA EXAMINADORA

Prof. André Luiz Avelino Sobral – MSc.

COORDENADOR

Prof. Rogério Malheiros dos Santos – DSc.

CONVIDADO

Prof. Manuel Martins Filho – DSc.

ORIENTADOR

Rio de Janeiro, 02 de Dezembro de 2021.

Dedico este trabalho a todos aqueles que me fizeram evoluir com o passar dos anos e aos meus pais que sempre estiveram me apoiando nos momentos de maior dificuldade.

AGRADECIMENTOS

Em primeiro lugar, agradeço a Deus por ter me permitido chegar a esse ponto, cuidando de mim e me guiando ao longo dos meus anos de estudo. Também agradeço a minha família que sempre esteve comigo e me ajudou no meu desenvolvimento pessoal, mais especificamente Mãe, Pai e Irmão.

Agradeço ao meu professor, orientador e amigo, Manuel Martins por ter me ajudado no desenvolver deste trabalho e ter me guiado durante minha jornada de graduação na faculdade.

Agradeço grandemente aos meus amigos que me incentivaram a continuar fortemente no desenvolvimento do TCC, removendo de mim pensamentos de desistência.

Gratifico também a todos aqueles que estiveram comigo enquanto cursava as matérias da faculdade, sejam eles alunos companheiros de turma ou professores que me ensinaram os mais diversos conceitos e engrandeceram minha formação acadêmica.

"Utilizar o jogo na educação infantil significa transportar para o campo do ensino-aprendizagem condições para maximizar a construção do conhecimento, introduzindo as propriedades do lúdico, do prazer, da capacidade de iniciação e ação ativa e motivadora."

(Kishimoto)

RESUMO

Na época em que vivemos, novas tecnologias vêm sendo constantemente desenvolvidas com o objetivo de solucionar e atender às diversas demandas de um cotidiano frenético e em constante mutação. Como resultado desse contexto, o espaço do conhecimento humano se alarga e deve ser passado de geração em geração. Isso acontece com frequência também nas grandes empresas, onde a Gestão do Conhecimento ocupa a primeira prateleira dos problemas a serem resolvidos. No entanto, com tantas áreas novas da ciência surgindo, o atual sistema de educação se mostra obsoleto na medida em que não acompanha, pelo menos como deveria, as mudanças nas relações sociais e, em especial, na utilização de forma efetiva dos recursos ofertados de forma generosa pela tecnologia. A educação brasileira ocupa posições acanhadas e até medíocres mesmo quando comparada com países com igual nível de desenvolvimento econômico e preenchemos sempre as últimas posições nos rankings de avaliação como no PISA. Além de uma política educacional mais adequada, moderna e consistente existe a necessidade da adoção de novas estratégias pedagógicas. O presente trabalho, tem por objetivo comentar sobre as vantagens da utilização de jogos como uma ferramenta educacional, expondo dados sobre a história geral dos jogos e seus benefícios ao desenvolvimento individual, assim como apresentar um projeto de jogo que foi desenvolvido com o objetivo de ser utilizado como uma ferramenta educativa para o aprendizado de lógica de programação, sendo instintivamente fácil de ser aplicada.

Palavras-chave: Jogos, Educação, Tecnologia.

ABSTRACT

In the times we live in, new technologies are constantly being developed with the objective of solving and meeting the diverse demands of a frenetic and constantly changing daily life. As a result of this context, the space for human knowledge expands and must be passed from generation to generation. This happens frequently also in large companies, where Knowledge Management occupies the first shelf of problems to be solved. However, with so many new areas of science emerging, the current education system is obsolete in that it does not follow, at least as it should, the changes in social relations and in the effective use of the resources offered generously by technology. Brazilian education occupies timid and even mediocre positions even when compared to countries with the same level of economic development and we always fill the last positions in the evaluation rankings as in PISA. In addition to a more adequate, modern, and consistent educational policy, there is a need to adopt new pedagogical strategies. This paper aims to comment on the advantages of using games as an educational tool, exposing data on the general history of games and their benefits to individual development, as well as presenting a game project that was developed with the aim of being used as an educational tool for learning programming logic, being instinctively easy to apply.

Keywords: Games, Education, Technology.

LISTA DE FIGURAS

Figura 1 — Jogo Real de Ur em exposição no Museu.	16
Figura 2 — Tabuleiro de Senet.....	17
Figura 3 — Tabuleiro de Mancala.....	19
Figura 4 — Tabuleiro de War.....	20
Figura 5 — Sistema de recompensa na Gamificação.	21
Figura 6 — Brinquedo infantil de encaixar formas.	23
Figura 7 — Tabuleiro de Xadrez.....	24
Figura 8 — Geração Z e a tecnologia.....	25
Figura 9 — Console Atari 7800	26
Figura 10 — Representação do jogo Pong, um dos primeiros jogos eletrônicos.....	27
Figura 11 — Tela do jogo Fábrica de Palavras	27
Figura 12 — Imagem do Robô Cody	28
Figura 13 — Exemplo de Código em GML	29
Figura 14 — Fluxograma de um Algoritmo.....	32
Figura 15 — Simbologia básica de um Fluxograma.....	33
Figura 16 — Linguagens de programação diversas.	34
Figura 17 — Estruturas representadas em C e sua saída.....	35
Figura 18 — Informações de um nível.....	36
Figura 19 — Tela de Criação de Nível.....	37
Figura 20 — Tela de Definições de Nível.....	38
Figura 21 — Tela de Níveis Criados.	39
Figura 22 — Exemplo de Código de Nível Válido.	39
Figura 23 — Funcionamento do CPU Custom.....	41
Figura 24 — Exemplo de objetivos de um nível.....	43
Figura 25 — Foto de memória RAM.	44
Figura 26 — Memória RAM representada dentro do projeto.	45
Figura 27 — Foto de um disco rígido.....	46
Figura 28 — Imagem de um processador AMD.	47
Figura 29 — CPU representado dentro do projeto.	47

LISTA DE SIGLAS E ABREVIATURAS

CPU	Central Processing Unit
GML	Game Maker Language
GMS2	Game Maker Studio 2
HD	Hard Disk
HTML	Hypertext Markup Language
RAM	Random Access Memory

SUMÁRIO

1	INTRODUÇÃO	13
1.1	Problema	13
1.2	Proposta.....	14
2	A IMPORTÂNCIA DOS JOGOS.....	15
2.1	Definindo o que é um jogo	15
2.2	Os primeiros jogos conhecidos	15
2.2.1	O jogo Real de Ur	16
2.2.2	Senet, a viagem ao outro mundo	17
2.2.3	Mancala e a semeadura.....	18
2.3	Jogos como representação da realidade	19
2.4	Gamificação, uma tendência moderna	20
3	ESTIMULANDO O DESENVOLVIMENTO NA INFÂNCIA	23
3.1	Ampliando o raciocínio	24
3.2	A geração Z	25
3.3	Jogos Digitais Educacionais.....	26
4	CODBOT- APRENDENDO PROGRAMAÇÃO SIMPLIFICADA	28
4.1	O que é um Algoritmo.....	29
4.1.1	Estruturas de controle.....	31
4.1.2	Representando um Algoritmo	31
4.1.3	Linguagens de programação	33
4.2	O sistema em sí	35
4.2.1	Aprenda com os níveis básicos	36
4.2.2	Crie o seu desafio.....	37
4.2.3	Objetivos e Placar.....	42
4.3	Conceitos utilizados no projeto	43
4.3.1	A Memória RAM.....	44
4.3.2	O HD.....	45
4.3.3	A CPU.....	46
5	CONCLUSÃO	48
	PROJETOS FUTUROS.....	48
	REFERÊNCIAS	49

ANEXO A – MANUAL DE COMANDOS.....	51
ANEXO B – MANUAL DE DESAFIOS.....	57

1 INTRODUÇÃO

Tendo em vista o grande avanço tecnológico que estamos presenciando nos últimos anos, com novas áreas do conhecimento surgindo em profusão, precisamos cada vez mais de uma educação mais eficiente para conseguir amparar e atender essa diversidade de conhecimento que vão sendo adquiridos e descobertos ao passar dos anos. Frequentemente, esse estoque de conhecimentos considerados inovadores acaba por se tornarem essenciais na vida cotidiana, sendo necessário introduzi-los no vetor de conhecimento como habilidades básicas e fundamentais.

Grande parte das pessoas ao ser apresentada a conceitos que nunca tiveram contato ou que não estejam familiarizados e habituados, tendem a não conseguir absorver o mínimo desejável e necessário sobre o tema, tendo grande dificuldade de adaptação.

De forma geral, um dos grandes desafios da atualidade se encontra na área de ensino, e um dos maiores problemas é a dificuldade de estimular os jovens a terem o seu próprio pensamento, o seu próprio jeito de formular novas ideias e de organizar suas abstrações mentais que promove naturalmente o seu desenvolvimento intelectual.

Diversos países como por exemplo Austrália, Estados Unidos e Japão já se deram conta da importância e dos benefícios do ensino de lógica de programação na infância em relação a isso, tanto que começaram a inserir na grade curricular básica das suas escolas, disciplinas desse gênero, objetivando buscar um grande desenvolvimento criativo e ampliar o seu desenvolvimento lógico, dessa forma inovando o seu sistema educacional em relação as técnicas de ensino já antes consolidadas.

1.1 Problema

O sistema de ensino em que estamos inseridos, não foi projetado para nos introduzir de forma eficiente no meio tecnológico, visto que praticamente a maior parte dele foi estruturada antes dessa grande explosão de uso de tecnologia. Em decorrência muitos jovens acabam tendo um déficit de conhecimento, em especial nas disciplinas de exatas como, por exemplo, matemática, estatística, física e informática em geral, onde conhecimentos de lógica são empregados na modelagem de problemas e na busca de uma solução eficaz e consistente.

Na realidade, o que aprendemos não é uma forma competente de pensarmos por nós mesmos, mas sim um modo de pensarmos sobre algo que já foi pensado antes por alguém que sabia como formular as próprias ideias e organizar o seu pensamento em busca de soluções para os problemas que foram vivenciados em sua época. Essa habilidade de como pensar individualmente e projetar os próprios conceitos e soluções não nos é ensinado. Dentro desse paradigma apenas em alguns poucos casos de fato, se consegue ter esse aproveitamento mental consistente que tanto é buscado nas redes de ensino do nosso país. Na maior parte dos casos, apenas o conhecimento superficial sobre um determinado assunto consegue ser propagado, não se desenvolvendo um conhecimento mais complexo sobre o tema em questão pela ausência de um processo reflexo mais apurado.

1.2 Proposta

Ao se avaliar o sistema de ensino atual, percebe-se que uma mudança ou uma nova adaptação deve ser feita com relação aos meios de introduzir conhecimento tecnológico, assim, para que novos conhecimentos venham a ser mais facilmente absorvidos pelos estudantes de uma forma mais intuitiva e natural, não basta apenas ensinar conteúdos novos usando métodos antigos. Dessa forma, qualquer que seja o conteúdo ele deve ser ensinado de forma inovadora tendo como objetivo principal facilitar o aprendizado.

Como esse propósito é desejável que novas técnicas sejam elaboradas para introduzir no nosso sistema de ensino meios mais eficientes de instruir os jovens, em especial os conceitos mais complexos, transformando o conteúdo em algo mais atrativo e lúdico. Além de comentar de maneira geral sobre jogos e a sua influência no desenvolvimento lógico, pretende-se com este trabalho, apresentar um projeto de jogo desenvolvido para ser utilizado como uma ferramenta básica de ensino de lógica de programação, de uma maneira que seja tão intuitiva como, por exemplo, montar um quebra-cabeças arrastando peças. A ideia é utilizar mecanismos que são amplamente conhecidos pelos jovens (nos jogos por exemplo) em parceria com os conceitos das áreas de tecnologia, para que sem maiores complicações os conteúdos possam ser assimilados naturalmente, trazendo benefícios no pensamento lógico e na facilidade de formular soluções aos desafios apresentados.

2 A IMPORTÂNCIA DOS JOGOS

2.1 Definindo o que é um jogo

Para se começar a falar sobre jogos, primeiramente temos que desenvolver um conceito básico sobre o que são jogos e os seus principais fundamentos, englobando aqui jogos em geral, tanto jogos físicos como aqueles que utilizam tabuleiros e cartas, até jogos digitais que necessitam, por exemplo de controladores e tecnologia para seu funcionamento.

O termo em questão tem origem no latim "*jocus*" que significa graça, brincadeira e divertimento, são quaisquer atividades em que existam pelo menos um ou mais jogadores, ou seja, aqueles que praticam tal atividade, e claro, um conjunto de regras bem definidas que devem ser obedecidas pelos envolvidos a fim de se completar um determinado objetivo. De modo geral, um jogo deve possuir:

- Um ou mais participantes;
- Um objetivo principal;
- Regras para explicar seu funcionamento;
- Condições para sucesso ou fracasso.

Essas são as características mais comuns e necessárias para se fundamentar um jogo válido. Existem dezenas de outras características para serem utilizadas em conjunto com essas, que podem tornar um jogo tão complexo como qualquer outra atividade, que demande extrema concentração e percepção do que está sendo realizado, ou simplesmente ser uma atividade completamente simples e divertida apenas pelo próprio prazer de se participar da mesma.

2.2 Os primeiros jogos conhecidos

A presença dos jogos é indiferente de cultura ou de época, as mais diversas culturas possuem seus próprios jogos e brincadeiras com suas próprias regras de participação detalhadas. Desde a antiguidade, os jogos vêm sendo criados como atividades recreativas, fazendo com que a sobrevivência se torne algo menos cansativo e exaustivo, dando novos sentidos à existência além das habituais práticas antigas.

Vamos considerar três de alguns dos jogos mais antigos: o jogo Real de Ur, o Senet e o Mancala, que são praticamente desconhecidos por quase toda a população atual. Na sequência falaremos um pouco sobre suas origens e mecânicas comentando algumas regras e o objetivo.

2.2.1 O jogo Real de Ur

Este é um jogo tão antigo, que suas regras originais são desconhecidas e a sua forma de jogar atual foi baseada em antigos documentos escritos, figuras e em outros jogos semelhantes a este. Ele foi encontrado em escavações em uma antiga cidade-estado de Ur (Iraque), por volta da década de 1920. Algumas empresas brasileiras fabricam e disponibilizam no mercado suas próprias versões desse jogo.

Sua mecânica é o de se executar um percurso ao longo do tabuleiro, semelhante a uma corrida com dois participantes rivais, o jogador deve chegar ao final o mais rápido possível com todas as suas peças para poder ganhar o jogo antes que o seu rival o faça. Através da rolagem dos dados é obtido o número de casas que o jogador poderá mover uma de suas peças e avançar pelo tabuleiro. De forma geral, se assemelha muito a um outro jogo bastante conhecido criado muito tempo depois chamado Pachisi posteriormente conhecido como Ludo.

Figura 1 — Jogo Real de Ur em exposição no Museu.



Fonte: Ludopedia (2017) <https://www.ludopedia.com.br/topico/16690/sobrevivendo-aos-anos-o-jogo-real-de-ur>

Seus dados são triangulares com 4 lados cada, sendo que, 2 de seus vértices são marcados com um ponto, sua contagem de valor é um pouco diferente do normal, ao serem lançados, para cada vértice marcado para cima se obtém 1 número, excepcionalmente, se nenhum dos 3 dados caírem com um vértice marcado para cima, obtém-se os 4 números, ou seja, para cada lançamento de dados pode-se andar entre 1 até 4 casas por vez.

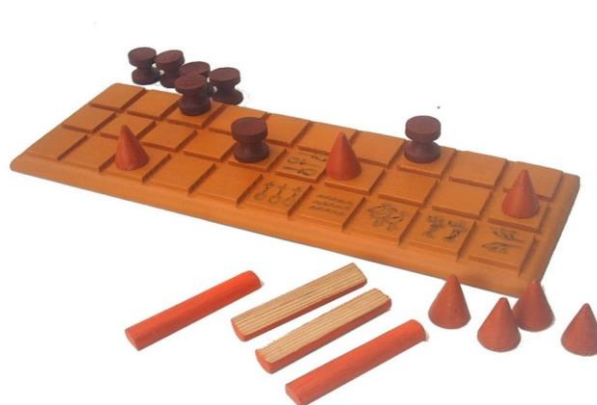
O tabuleiro é dividido em 2 percursos, sendo que cada jogador possui o seu próprio, porém o caminho do meio é compartilhado entre os dois jogadores, sendo esse onde pode ocorrer a captura das peças rivais, que acontece quando um jogador cai na casa em que seu adversário está alocado. a peça deve então ser retirada do tabuleiro e retornada ao seu dono para recomençar todo o percurso com a mesma novamente, caso uma peça caia em uma casa com uma roseta desenhada nela, o jogador ganha uma jogada extra e a peça que está alocada nela, não pode ser capturada pelo seu oponente.

2.2.2 Senet, a viagem ao outro mundo

Outro jogo de tabuleiro dos mais antigos que se tem registro, extremamente popular no Antigo Egito, tanto que nas mais diversas escavações realizadas, foram encontrados vários tabuleiros diversos de Senet e variadas figuras de jogadores de Senet que estavam ilustradas em desenhos antigos na parede.

Embora esse jogo tenha sido utilizado como forma recreativa, o Senet tem um significado religioso para os egípcios, representando uma viagem da alma através do outro mundo enfrentando pelo caminho as forças maléficas e os inimigos do deus dos mortos Osíris, o nome Senet tem como uma tradução “Passagem”.

Figura 2 — Tabuleiro de Senet



Os dados do jogo são 4 palitos com 2 lados diferentes, sendo um lado com a cor preta ou marrom, eles devem ser segurados na forma vertical e então largados para poder realizar a contagem do número, cada lado preto para cima resulta em 1 número obtido e caso todos os palitos tenham o lado marrom revelado para cima, o número resultante será 6, ou seja, o número que pode ser obtido vai de 1 até 6, excluindo o número 5.

O tabuleiro possui uma grade com 30 casas, sendo 3 na vertical e 10 na horizontal, possuindo algumas casas com regras especiais, as peças devem ser posicionadas na primeira linha horizontal do tabuleiro, opostas as casas especiais, sendo as peças brancas nas casas ímpares e as pretas nas casas pares, sendo o movimento das peças realizados em ziguezague.

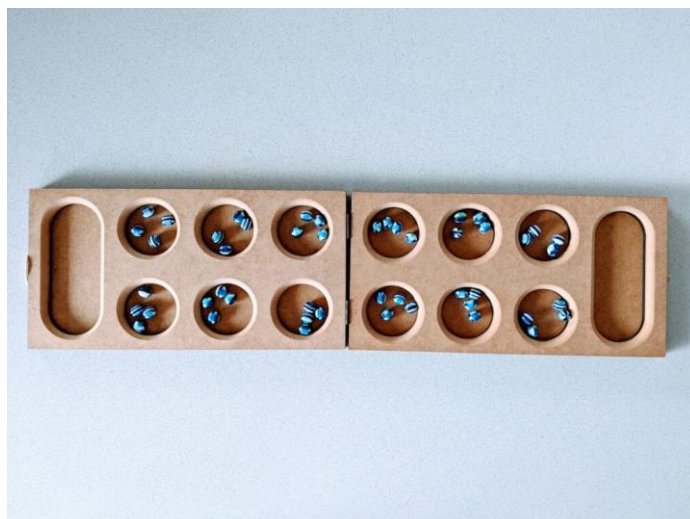
Seu objetivo é igual ao do Jogo Real de Ur, o jogador deve ultrapassar todas as suas peças até a linha de chegada antes de seu oponente para ganhar o jogo, mas Senet possui uma dificuldade um pouco maior de ser aprendido pois possui mais tipos de regras que tornam suas partidas mais complexas de se entender e com a diversidade de casas especiais as partidas se tornam mais pensativas e um tanto estratégicas.

2.2.3 Mancala e a semeadura

Mancala possui uma ambientação de plantio, em seu tabuleiro tem-se a representação da semeadura, sendo que as peças são as sementes e cada casa possui um número fixo de sementes colocada nela no começo de cada partida, possuindo um total de 6 casas no tabuleiro, cada jogador possui seu próprio oásis (também chamados de *kalah*), que é a área que devem ser colocadas as colheitas para obtenção da pontuação durante a partida e se iniciam vazias, sem nenhuma semente colocadas nela.

As rodadas vão sendo alternadas entre os jogadores, que devem no seu turno escolher uma das suas casas e recolher todas as sementes que estão colocadas nela para então replantar em outras casas, começando a distribuição delas indo no sentido da sua direita, plantando uma semente por casa, plantando também no seu oásis e nas casas adversárias, mas nunca no oásis do adversário. O jogador ganha uma nova jogada caso a última semente que ele plantar acabe caindo no seu próprio oásis.

Figura 3 — Tabuleiro de Mancala



Fonte: Criandocomapego (2021) <https://www.criandocomapego.com/mancala-o-que-e-como-se-joga-e-objetivos/>

Há duas formas de se obter sementes, as que são colocadas diretamente no oásis e as que são capturadas do adversário. Para capturar sementes do adversário, a última semente plantada deve cair em uma casa vazia do jogador que a plantou, então as sementes dos adversários na casa logo oposta e a mesma são automaticamente levadas para o oásis do jogador atual.

Para finalizar uma partida, todas as 6 casas de um dos jogadores devem ficar livre de sementes, ou seja, um dos lados do tabuleiro tem que ficar vazio, excluindo o seu oásis, todas as peças restantes são colocadas no oásis do seu oponente, exemplificando, se o jogador A tiver suas casas sem nenhuma semente, então todas as sementes que estiverem nas casas do jogador B vão para o oásis do jogador B e então a partida é finalizada e a pontuação contabilizada, ganha-se a partida o jogador que estiver com o número maior de sementes no seu oásis.

2.3 Jogos como representação da realidade

Como visto nos jogos apresentados, nem todos são apenas recreativos ou focados no lazer, uma parte considerável deles tem uma história de fundo, ambientada em alguma situação da realidade em que foram pensados e representados através de um tabuleiro ou outro sistema físico. Sendo assim, são mais do que apenas um conjunto de regras e peças, podendo contar histórias de outras culturas através de partidas, podendo até ser considerados artefatos históricos de determinada cultura.

Outro jogo que é baseado na realidade, temos o jogo de tabuleiro War, cujo um dos objetivos existentes é a conquista de territórios ao redor do mundo ou a destruição de exércitos rivais.

Figura 4 — Tabuleiro de War



Fonte: Ludopedia (2021) <https://www.ludopedia.com.br/jogo/war>

Dessa forma percebe-se que representar o mundo real através de um jogo, o torna algo muito mais interessante e o jogador acaba por aprender conceitos novos e reais através de uma simples partida de um jogo descontraído. Tendo isso como base, por que não envolver mais os jogos com a realidade e tentar aplicar novos conhecimentos como uma forma não apenas de lazer mas também de aprendizado? Ao pensar em novas formas de disseminar o conhecimento vemos que a combinação de uma atividade intelectual com algo prazeroso torna essa prática menos cansativa e atrativa ao praticante.

2.4 Gamificação, uma tendência moderna

Quando se trata de jogos, qualquer atividade pode ser mais atrativa e descontraída quando transformada em um jogo. Essa nova prática é chamada de *Gamification*, ou Gamificação em português, também conhecida como Ludificação.

A gamificação é a utilização de técnicas de incentivo baseadas em mecânicas de jogos em atividades que não tem uma relação direta com jogos, ou seja, literalmente transformar algo parcialmente em um jogo, da melhor forma possível, para que tal atividade seja mais atrativa, emocionante e produtiva para quem a realiza. Pretende-se assim, estimular o indivíduo através de desafios e/ou recompensas para que se consiga executar atividades complexas de forma mais descontraída.

Uma gamificação contém usualmente os seguintes conceitos:

- Conquistas de diferentes níveis que funcionam como missões que devem ser realizadas pelo jogador;
- Uma espécie de classificação, para que cada vez que uma conquista seja corretamente realizada o jogador melhore o seu placar atual;
- Um sistema de loja, para que os seus pontos possam ser trocados por recompensas e demais incentivos;
- Possibilidade de cooperação entre os participantes, sendo possível trocar itens, recursos ou pontuações entre os mesmos;
- Desafios competitivos caso o sistema em questão possua diversos participantes, para que se sintam mais entusiasmados a superarem as conquistas.

Figura 5 — Sistema de recompensa na Gamificação.



Fonte: Ludospro (2021) <https://www.ludospro.com.br/blog/o-que-e-gamificacao>

Atualmente a gamificação é uma estratégia muito aplicada nas empresas para se impulsionar a produtividade dos seus colaboradores, visto que sua utilização traz vários benefícios distintos, tais como aumento da motivação e produção dos envolvidos, redução de estresse, melhoria da comunicação interna, desenvolvimento de habilidades dos funcionários entre outros.

Engana-se quem pensa que a gamificação só pode ser utilizada no âmbito empresarial, já que ela pode ser aplicada nas mais diversas áreas, inclusive como uma técnica educacional para estimular o aprendizado entre jovens que são mais acostumados com esse tipo de abordagem rompendo o antigo padrão de aprendizagem vigente. Uma grande parte deles, já conhece os conceitos da gamificação e levam menos tempo para se acostumar com a nova abordagem de aprendizado, pois é nessa faixa etária que os jogos são mais comuns e estão mais presentes no seu cotidiano. Suas dinâmicas de aprendizado se resumem em grande parte a missões e desafios entre os estudantes para estimulá-los a completar objetivos das atividades através de um sistema para obtenção de pontuação.

3 ESTIMULANDO O DESENVOLVIMENTO NA INFÂNCIA

A infância é uma das etapas fundamentais para a formação de cada indivíduo, esse é o período de desenvolvimento que engloba o nascimento até a puberdade, sendo considerado por muitos até os doze anos de idade. Iremos dividir essa fase em 3 partes, sendo elas:

- Primeira Infância (0-3 anos): Onde acontece os primeiros desenvolvimentos cerebrais, ocorrendo um grande desenvolvimento emocional, é nessa fase que ocorre o desenvolvimento das aptidões básicas que serão futuramente necessárias para aprimorar habilidades mais complexas;
- Segunda Infância (3-6 anos): Aqui tem-se um aumento de suas forças e das habilidades motoras. Ocorre desenvolvimento na sua imaginação e criatividade, aumento da independência e do autocontrole;
- Terceira Infância (6-12 anos): Já nessa terceira parte, obtém-se um desenvolvimento maior do pensamento, onde se começa a pensar de forma mais lógica e estimular o lado cognitivo.

A partir da terceira infância em diante ou até pouco antes do fim da segunda, atividades que demandam pequena quantidade de pensamento, tornam-se mais recomendadas de serem praticadas pelo indivíduo. Muitas crianças começam a solucionar pequenos problemas já no fim da primeira infância com alguns brinquedos mais simples, como os clássicos de encaixar formas geométricas em buracos específicos para o mesmo com quadrados, triângulos, círculos, retângulos entre outras.

Figura 6 — Brinquedo infantil de encaixar formas.



3.1 Ampliando o raciocínio

Muitas pessoas acabam por se desenvolver através de jogos, seja fisicamente ou mentalmente, mesmo que muitas vezes não de forma intencional, pois diversos jogos exigem uma certa quantidade de raciocínio em suas partidas. Jogos como quebra-cabeças por exemplo, que são focados na resolução de um dado problema, utiliza-se bastante do raciocínio e pode por sua vez melhorar a percepção e aumentar a capacidade na resolução de problemas.

Falando sobre jogos que desenvolvem o raciocínio, temos um dos mais famosos jogos de tabuleiros, o Xadrez, um jogo tão complexo que possui milhares de possibilidades de jogadas.

Figura 7 — Tabuleiro de Xadrez



Fonte: Brasilescola (2021) <https://brasilescola.uol.com.br/educacao-fisica/xadrez.htm>

O xadrez quando jogado com frequência, estimula a concentração e a tomada de decisões, sendo que cada movimento ao desenrolar da partida possui seus riscos que devem ser avaliados. Muitas escolas utilizam o xadrez como uma ferramenta educacional para os seus alunos, incentivando a sua prática desde a infância como forma de desenvolver o pensamento lógico e o raciocínio estratégico.

3.2 A geração Z

Estamos rodeados das mais diversas inovações tecnológicas, assim, em nosso cotidiano novidades surgem quase que diariamente para atender demandas e com isso gerações diferentes de indivíduos acabam sendo moldadas pela época em que nasceram. Se pararmos para pensar nas décadas anteriores, quando as tecnologias começaram a ser desenvolvidas e lançadas no mercado, grande parte dos adultos não conseguiam se adaptar aos novos equipamentos desenvolvidos. Agora, no entanto, por já nascerem imersas nesse ambiente frenético recheado de novidades tecnológicas as crianças conseguem desenvolver rapidamente fluência na sua utilização em um tempo às vezes imperceptível para os adultos.

A chamada geração Z é formada pelos indivíduos que nasceram a partir da metade da década de 90, sendo sucessora da geração Y, e que possuem uma forte conexão com a tecnologia. Durante esses anos, houve grandes inovações tecnológicas e indivíduos dessa geração acabaram por crescer acostumados com todas essas novas mudanças e desenvolveram um forte apego tecnológico de uma forma tão natural como começar a andar ou a falar.

Figura 8 — Geração Z e a tecnologia



Fonte: Diariodocomercio (2019) <https://diariodocomercio.com.br/gestao/geracao-z-esta-alem-de-estereotipos/>

Nascendo em um ambiente totalmente tecnológico sua adaptação foi feita de uma forma suave e sem maiores dificuldades, assim, hoje em dia, praticamente a maioria dos jovens da geração Z são fluentes na utilização das tecnologias mais modernas e conseguem se adequar a novos ajustes sem maiores problemas. Para eles, a tecnologia se tornou algo, de certa forma,

mais natural, pois estão em constante vivência com o mesmo, sendo assim, por que não tornar essa prática algo mais benéfico para o desenvolvimento dos jovens?

Para essa tarefa novas ideias para unir tecnologia e aprendizado devem ser pensadas e uma ótima forma de se fazer isso, é reunir as coisas que mais atraem o interesse do público com aquilo que se quer ser ensinado, através da tecnologia utilizando-se como meio principal os jogos.

3.3 Jogos Digitais Educacionais

Um jogo digital também chamado de videojogo, é um jogo eletrônico, ou seja, jogado por meios digitais, necessitando de algum tipo de tecnologia para poder ser efetivamente jogado, normalmente essa tecnologia se baseia em um console (um computador feito para jogos) e seus controladores mas não apenas a isso. Hoje em dia, a variedade de jogos digitais vem crescendo de forma extremamente rápida e a sua utilização se torna cada vez mais comum e frequente entre os jovens como um passatempo ou até mesmo carreira profissional.

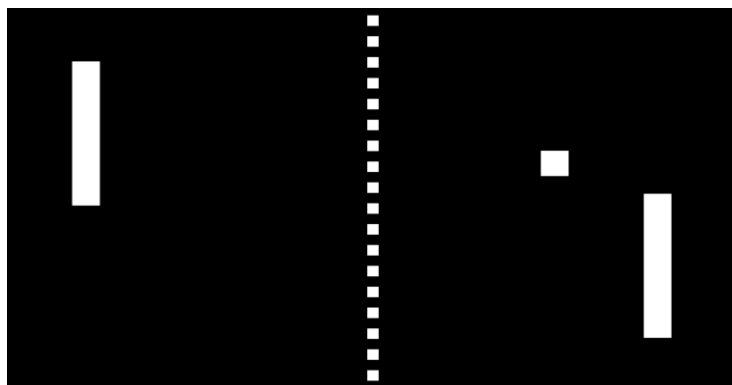
Figura 9 — Console Atari 7800



Fonte: Wikipédia (2016) https://pt.wikipedia.org/wiki/Atari_7800

Os primeiros jogos digitais conhecidos eram bastante simples por causa da limitação de hardware que se possuía naquela época, mas com a evolução das tecnologias de hardware e software disponíveis, o desenvolvimento de jogos digitais se tornou algo mais simples e prático, com resultados mais agradáveis e utilização de algoritmos extremamente avançados.

Figura 10 — Representação do jogo Pong, um dos primeiros jogos eletrônicos



Fonte: Super Interessante (2016) <https://super.abril.com.br/blog/oraculo/quem-inventou-o-videogame/>

Agora pensando no sentido de jogos digitais educacionais, temos em evidência o propósito dos jogos, que são, como o nome diz, jogos digitais criados com o intuito de serem usados como ferramentas na educação. Como já sabemos, jogos possuem uma forte influência no desenvolvimento das pessoas que os utilizam, sendo ela positiva ou às vezes até negativa. Baseando-se nessa premissa, foram criados diversos jogos com temáticas educativas para jovens e crianças para facilitar o ensinamento de determinado tema.

Figura 11 — Tela do jogo Fábrica de Palavras



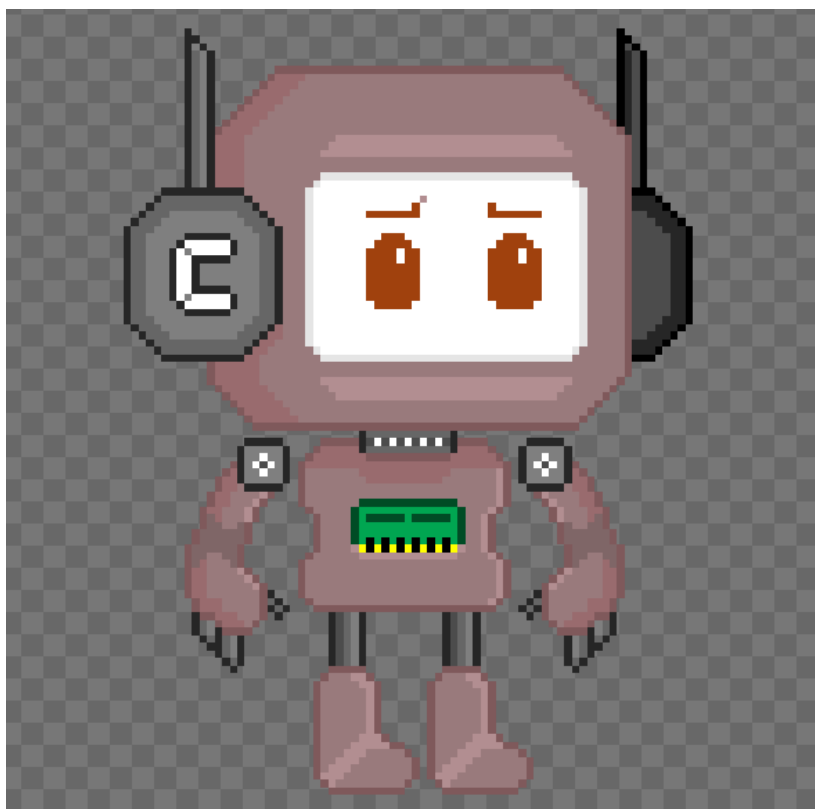
Fonte: Utilizandomidias (2011) <https://utilizandomidias.blogspot.com/2011/05/jogos-educativos-de-portugues.html>

Muitos jogos educativos possuem como tema o ensino de palavras de um idioma, operações básicas de matemática, raciocínio e associações por imagem ou alimentação correta para crianças. Podem também abordar temas mais avançados e complexos como como no ensino de cálculo, física, química ou outras ciências em geral para adolescentes. Através da representação simplificada daquilo que se quer ensinar, busca-se facilitar a absorção de novos conteúdos utilizando-se de estratégias pensadas nos jogos.

4 CODBOT- APRENDENDO PROGRAMAÇÃO SIMPLIFICADA

Ao se desenvolver um projeto, deve-se ter em mente aquilo que se deseja alcançar, ou seja o objetivo, que no caso, é a facilitação do aprendizado de lógica de programação, através de um jogo digital de forma simplificada. Pensando dessa forma, foi desenvolvido o CodBot, para ser um amigo daqueles que desejam aprender ou ensinar programação de forma intuitiva e divertida, seja para crianças ou adolescentes.

Figura 12 — Imagem do Robô Cody



Fonte: O Autor.

O conceito de programação, se baseia em escrever programas de computador usando códigos, também chamados de algoritmos. Um algoritmo é uma sequência de ações que se executadas realizam determinada tarefa. Para que o algoritmo possa ser executado em uma máquina (computador) ele precisa ser escrito em uma linguagem de programação específica, ou seja, em comandos que o computador entende e pode executar. Na verdade, um programa é um artifício que utilizamos para dizer ao computador o que queremos que ele faça.

Figura 13 — Exemplo de Código em GML

```
nota1=8
nota2=10
media=(nota1+nota2)/2

if media>=7
    show_message("Aprovado")
else
    show_message("Reprovado")
```

Fonte: O Autor.

O nosso projeto foi desenvolvido utilizando o IDE GMS2 (Game Maker Studio 2) e a sua linguagem de programação nativa GML (Game Maker Language), estando disponível para duas plataformas atualmente, sendo elas um Executável do Windows ([GitHub](#)) e uma Aplicação Android ([GitHub](#) e [GooglePlay](#)).

4.1 O que é um Algoritmo

Quando se começa a descrever o funcionamento de um processo, como por exemplo uma receita, um procedimento ou um programa de computador, um conjunto básico de ações, chamadas de instruções primitivas, devem ser conhecidas e passíveis de serem executadas pelo responsável do processo, o mesmo vale para cozinheiros, assistentes administrativos e até mesmo computadores.

Entendido como um procedimento ou rotina, um algoritmo pode ser definido de algumas maneiras distintas, no entanto a mais adequada para o nosso propósito é a seguinte:

Uma sequência ordenada, finita e não ambígua de etapas que conduzem à solução de um problema.

Dessa forma, embora o conceito de algoritmo esteja intimamente relacionado ao domínio computacional, ele pode ser entendido com uma definição mais abrangente de ser um **processo**, sub-rotina ou procedimento e dessa forma, pode ser aplicado em diversos contextos, mesmo não computacionais.

Vamos considerar dois exemplos preliminares para esse entendimento. O primeiro é o exemplo clássico no contexto culinário de preparo de uma receita genérica.

Exemplo-1 - Contexto Culinário

Misture os ingredientes.

Unte o tabuleiro com manteiga.

Despeje a mistura no tabuleiro.

Se (*há queijo parmesão*) **Então**

Espalhe sobre a mistura.

Leve o tabuleiro no forno.

Enquanto (*não corar*).

Deixe o tabuleiro no forno.

Deixe esfriar.

Experimente antes de servir.

Exemplo-2 - Contexto Administrativo

Verifique preenchimento do formulário.

Se (*preenchimento correto*) **Então**

Arquive o documento.

Forneça protocolo.

Senão

Lamente.

Torne a lamentar.

Mande o cliente comprar outro formulário.

Despeça-se educadamente do cliente.

Aqui percebe-se que ações como **Unte o tabuleiro com manteiga** e **Leve o tabuleiro ao forno** são consideradas tarefas básicas do domínio de quem executa este processo: “o cozinheiro”, da mesma forma que as ações **Arquive o documento** e **Forneça protocolo** são de conhecimento do auxiliar administrativo.

Portanto, em todos os contextos considerados, as instruções primitivas são o passo inicial para criação de processos de maior complexidade – algoritmos mais sofisticados! No entanto, a forma de agregação das instruções é um aspecto que se aplica a todos os processos, independentemente do contexto ao qual se referem. Esses mecanismos são chamados de **estruturas de controle** e controlam como a execução de um processo é combinada com as instruções subjacentes que o constituem.

4.1.1 Estruturas de controle

Existem somente três formas de agregar instruções em um processo (algoritmo): sequência, seleção e repetição. Vamos inicialmente exemplificar as duas primeiras – sequência e seleção (condição).

Observando o exemplo-1 percebe-se que as ações são executadas **sequencialmente**, mas em uma **ordem** específica (não podemos despejar a mistura no tabuleiro sem antes misturar os ingredientes e untar o tabuleiro com manteiga)

Misture os ingredientes.

Unte o tabuleiro com manteiga.

Despeje a mistura no tabuleiro.

No entanto, no exemplo-1 a ação abaixo só será executada se a **Condição** (*há queijo parmezon*) for verdadeira:

Espalhe sobre a mistura.

Da mesma forma, no exemplo-2 (contexto administrativo) as ações abaixo só serão executadas se a **Condição** (*preenchimento correto*) for verdadeira:

Arquive o documento.

Forneça protocolo.

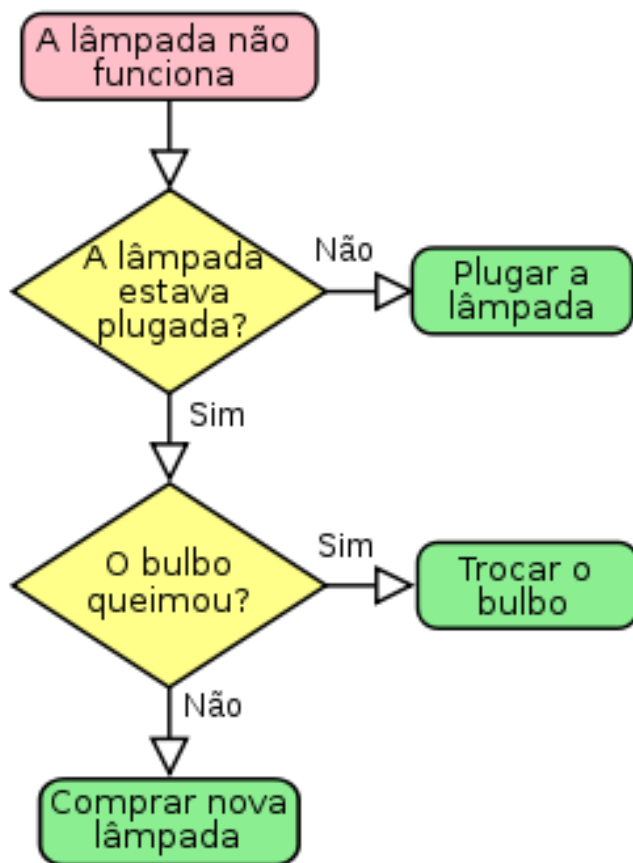
De posse desses conceitos iniciais, o CodBot começou a ser desenvolvido. Buscando simplificar e facilitar o processo de programação surgiu a ideia de utilizar comandos do tipo *Drag-and-drop* (arrastar e soltar). Assim, em um jogo de quebra-cabeças por exemplo, o jogador consegue programar o robô para realizar as diversas ações possíveis facilmente, vendo na prática a execução de cada uma delas em ordem sequencial.

4.1.2 Representando um Algoritmo

Alguns algoritmos tendem a ficar bastante complexos com o seu desenvolvimento tornando a leitura de seu código e sua compreensão difícil. Assim, para facilitar o entendimento e funcionamento dos algoritmos, alguns diagramas foram construídos. Nesse tópico, iremos abordar o diagrama chamado de fluxograma que é o mais utilizado para esse propósito.

Um fluxograma é um gráfico que indica todos os caminhos que podem ser percorridos por uma determinada estrutura, nesse caso o algoritmo, representando assim sua lógica de funcionamento e suas possíveis ações partindo do início de sua execução e chegando até o seu eventual término.

Figura 14 — Fluxograma de um Algoritmo



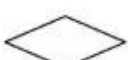






Fonte: Wikipédia (2021) <https://pt.wikipedia.org/wiki/Fluxograma>

Uma das vantagens da utilização de fluxogramas como representação de algoritmos é facilitar o entendimento do algoritmo através da representação visual do seu fluxo de execução de forma que ele possa ser entendido por outras pessoas mais intuitivamente.

Para se representar corretamente um algoritmo utilizando-se fluxogramas, deve-se conhecer inicialmente o conjunto de símbolos que são usados nessa representação.

Figura 15 — Simbologia básica de um Fluxograma

	Indica o início ou fim do processo
	Indica cada atividade que precisa ser executada
	Indica um ponto de tomada de decisão
	Indica a direção do fluxo
	Indica os documentos utilizados no processo
	Indica uma espera
	Indica que o fluxograma continua a partir desse ponto em outro círculo, com a mesma letra ou número, que aparece em seu interior

Fonte: Qualiex (2012) <https://blogdaqualidade.com.br/fluxograma-de-processo/>

Como mencionado anteriormente no projeto do *CodBot*, a construção dos algoritmos é feita utilizando-se um sistema de *Drag-and-drop* que evita a utilização de códigos escritos. Com esse sistema o jogador escolhe os comandos que vão ser utilizados e os arrasta criando uma ordem sequencial de passos a serem executados, isso gera um fluxograma com a sequência os comandos que serão executados.

4.1.3 Linguagens de programação

Uma linguagem de programação é o instrumento de comunicação usado por um programador para comunicar ao computador os comandos e as instruções que devem ser executadas. As linguagens de programação permitem desenvolver programas utilizando-se símbolos, palavras-chaves, regras e outras estruturas pré-determinadas de cada linguagem específica.

Figura 16 — Linguagens de programação diversas.



Fonte: Auditest (2020) <https://auditeste.com.br/linguagens-programacao-estudar-2020/>

As linguagens, desde a mais antiga até a mais recente, possuem suas próprias características e peculiaridades que as diferenciam umas das outras. Cada uma foi construída com um determinado objetivo e função. Temos por exemplo, linguagens voltadas para Web como HTML, CSS e Javascript, linguagens para análise de dados como R, linguagens que podem ser utilizadas para desenvolvimento *mobile* como Kotlin, linguagens mais abrangentes como Python e Java, entre outras.

No entanto, todas essas linguagens possuem conceitos mais genéricos que abrangem a sua grande maioria, estruturas e ideias que são indispensáveis para o seu funcionamento. Iremos agora citar algumas importantes estruturas e fundamentos gerais de uma linguagem de programação, sendo elas:

- Variáveis: São como caixas de memória, servem para armazenar um determinado valor de um tipo específico de dado, como numérico, literal ou booleano;
- Estruturas de Controle: São tipos de estruturas que normalmente trabalham em conjunto com as variáveis, analisando o seu valor para então escolher um fluxo a se seguir ou comandos a se repetir;
- Modularização ou Subprograma: São trechos de código separados que podem ser rapidamente executados em diversas partes do código principal;

- Tipos Abstratos de Dados: Tipos de dados criados pelo próprio usuário para suprir sua necessidade que os tipos padrões não podem fazer, como por exemplo as pilhas, listas ou árvores.

Figura 17 — Estruturas representadas em C e sua saída.

```

1  #include <stdio.h>
2  #include <stdlib.h>
3
4  //Definição do subprograma
5  void subprograma(){
6      printf("Isso aqui é um trecho de código separado");
7  }
8
9  int main()
10 {
11     //Váriaveis
12     int inteiro = 4;
13     char letra = 'b';
14     float decimal=2.2;
15
16     //Pilha
17     struct Pilha{
18         int num;
19         struct Pilha *prox;
20     };
21     typedef struct Pilha pilha;
22     pilha *PILHA = (pilha *) malloc(sizeof(pilha));
23
24     //Estruturas de controle
25     if (inteiro==4)
26     else
27
28     for (inteiro;inteiro<10;inteiro++)
29
30     //Chamada do subprograma
31     subprograma();
32 }

```

É igual a Quatro
4 5 6 7 8 9 Isso aqui é um trecho de código separado

Fonte: O Autor (2021).

4.2 O sistema em si

Codbot é um jogo em estilo quebra-cabeças em que o objetivo principal é atravessar os níveis propostos utilizando-se de comandos específicos (arrastáveis) que são programados para o robô Cody executar. A cada nível completado novos níveis vão surgindo com novas mecânicas para se utilizar com o objetivo de alcançar o ponto de chegada, sendo este o objetivo principal dos desafios, que é uma bandeira verde presa no chão existente em cada desafio.

Com o avanço do jogador através dos níveis, comandos novos podem ser utilizados dificultando cada vez mais a resolução dos problemas propostos. Certos níveis possuem

múltiplas soluções e caminhos para serem utilizados dependendo apenas da criatividade do jogador para escolher a opção que preferir utilizar.

Utilizando uma metodologia de dificuldade progressiva, onde os primeiros níveis a se concluir são simples e os posteriores tendem a ser de maior dificuldade, o jogador tem o seu desenvolvimento estimulado de forma natural. Depois de alguns desafios resolvidos, o jogador consegue ir concluindo os níveis mais rapidamente até se acostumar com o sistema de *Drag-and-drop*, quando finalmente estará programando facilmente o Cody sem nem se dar conta de que o que está fazendo, é desenvolver um algoritmo. O projeto possui dois modos de jogo: os níveis básicos, que são os desafios já inclusos no sistema, e o criador de desafios, que são os níveis desenvolvidos pelos próprios usuários.

4.2.1 Aprenda com os níveis básicos

O principal objetivo dos níveis básicos, é ensinar os fundamentos e de todo o sistema de funcionamento do jogo de forma gradual até que todas as mecânicas sejam absorvidas pelo jogador. Começando com desafios extremamente simples que podem ser concluídos com comandos apenas de movimento, pretende-se ensinar a utilizar cada comando disponível no sistema ao se concluir e avançar os níveis.

Ou seja, foram construídos um total de 18 desafios iniciais que servem de tutorial para demonstrar na prática como algumas mecânicas são abordadas no projeto. Para cada desafio concluído um novo tipo de comando será abordado seguindo sequencialmente a ordem dos comandos disponíveis na CPU.

Figura 18 — Informações de um nível.



Fonte: O Autor (2021).

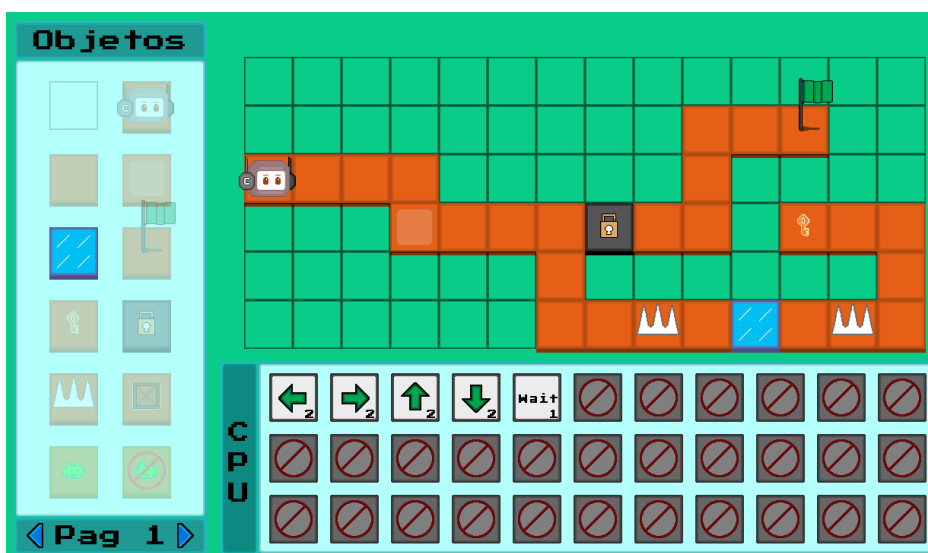
Para cada um desses níveis iniciais, uma quantidade limitada de comandos está disponível, para que o jogador perceba quais devem ser usados para concluir aquele desafio proposto. Cada desafio tem suas próprias características que podem ser acessadas assim que o nível é escolhido pelo jogador. Sendo elas:

- Título: uma pequena frase intuitiva que indica o tema geral daquele desafio;
- Descrição: contém normalmente um comentário ou uma dica de como o desafio pode ser resolvido;
- RAM: indica a quantidade máxima de memória que pode ser utilizada pelo jogador no nível específico para concluir o desafio;
- CPU: dita ao jogador as instruções disponíveis para serem utilizadas naquele nível;
- Placar: aponta o melhor resultado obtido pelo jogador naquele nível específico.

4.2.2 Crie o seu desafio

Para melhorar a diversidade dos quebra-cabeças dispostos no projeto, foi adicionado uma mecânica de editor de níveis, onde os jogadores podem criar seus próprios desafios e compartilhar com outras pessoas para que elas possam concluir uma maior diversidade de desafios. Dessa forma, um professor pode, por exemplo, optar por construir desafios próprios e enviar para seus alunos completarem.

Figura 19 — Tela de Criação de Nível.



Fonte: O Autor (2021).

Ao iniciar o modo de criação, é apresentado ao jogador uma matriz de 6x14 quadrados, que podem ser preenchidos com os objetos disponíveis à esquerda da tela, sendo disponibilizado quase todos os objetos do projeto para serem colocados no nível, o robô Cody, a bandeira de chegada, os itens, plataformas e outros. Alguns objetos são obrigatórios para que o nível possa ser corretamente jogável, como a bandeira e o jogador, caso contrário não poderá ser jogado até esses itens serem adicionados.

Um nível pode ser compartilhado com outros jogadores através do seu código, que pode ser obtido através das definições e então ser importado no projeto, na parte de dados posteriormente.

Figura 20 — Tela de Definições de Nível.



Fonte: O Autor (2021).

Uma quantidade limitada de níveis pode ser criada e armazenada por vez, sendo esse limite de 4 níveis, a partir disso não será possível criar mais níveis personalizados, sendo necessário editar um já existente ou então fazer a sua exclusão para liberar espaço para a criação de um outro novo desafio. Para se contornar essa situação, pode-se por exemplo salvar os códigos dos níveis para posteriormente os deixar recuperáveis sempre que for necessário, evitando assim a sua eventual perda.

Figura 21 — Tela de Níveis Criados.



Fonte: O Autor (2021).

Sempre que um nível é construído, ele é armazenado em um arquivo no formato TXT (arquivo de texto) que posteriormente pode ser lido pelo projeto que então representa o seu conteúdo como um nível jogável através de um código de 90 caracteres ou 100 caracteres caso o processador seja Custom (Processador que pode personalizar os comandos disponíveis).

Figura 22 — Exemplo de Código de Nível Válido.

```

AAAAA AAAA AAAA AAAA
AAAAA AAAA ACCFAA
BCCCA AAAA ACAAAA
AADCC CHCCAGCC
AAAAA ACAAAAAA AC
AAAAA ACCICECIC
QB0107

```

Fonte: O Autor (2021).

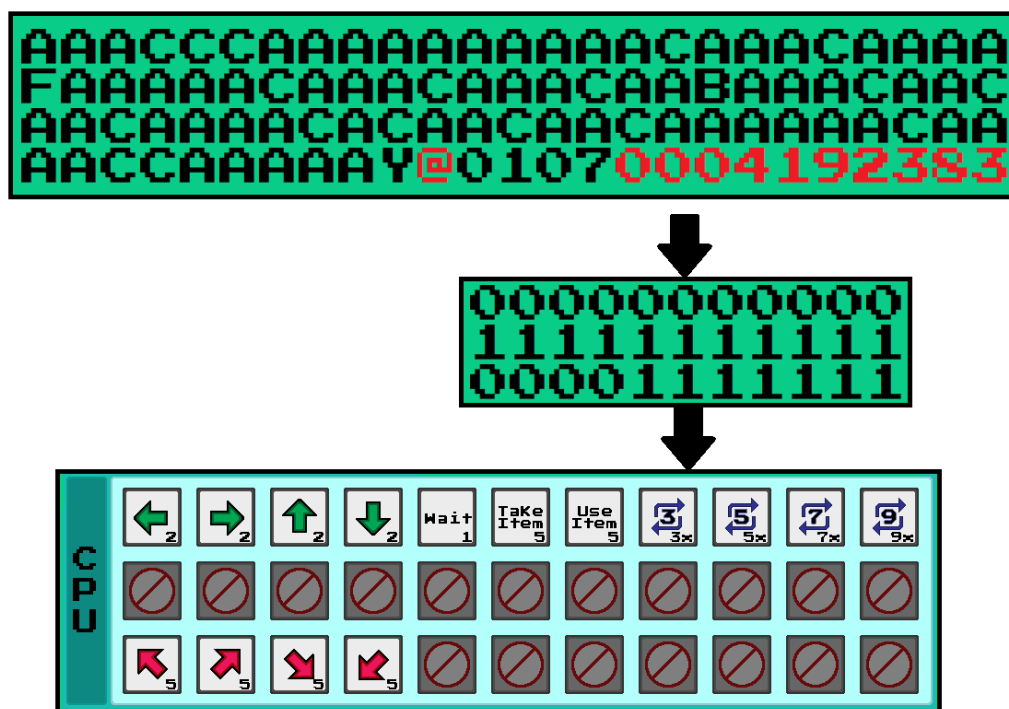
O código é validado e construído da seguinte forma: Os primeiros 84 caracteres, representam a matriz de 6x14 quadrados, sendo cada caractere o código do objeto naquela

posição. Por exemplo, a letra “A” representa um espaço vazio e a letra “B” a posição do robô Cody, já os últimos caracteres são códigos de configuração do nível, sendo eles:

- Caractere número 85: Representa a quantidade de memória RAM disponível no nível, na imagem temos “Q” representando 110 MB de RAM;
- Caractere número 86: Representa o processador disponível no nível, na imagem temos “B” representando o processador “Basic”;
- Caractere número 87 e 88: Representa o desafio A, na imagem temos “01” sendo o desafio "Não use comandos falhos";
- Caractere número 89 e 90: Representa o desafio B, na imagem temos “07” sendo o desafio "Use até 20 comandos no robô".

Os Caracteres de número 91 até 100 são mais especiais e específicos, eles são opcionais e só são usados caso o processador seja Custom (personalizado) cujo símbolo usado no código é “@”. Esses caracteres são na verdade um número inteiro de 10 dígitos, que ao ser lido é convertido em um número binário com 33 dígitos, representando os comandos que vão ser bloqueados (1) ou não bloqueados (0). O processador Custom é útil para caso as opções de comandos tenham que ser limitadas para um determinado nível e não tenha um processador equivalente ao que necessite ser utilizado. Por exemplo, um que o robô não possa andar para os lados, andando apenas diagonalmente, então o processador Custom deverá ser usado bloqueando os comandos de movimento normais.

Figura 23 — Funcionamento do CPU Custom.



Fonte: O Autor (2021).

Caso o código não siga esse determinado padrão, ele não poderá ser importado no projeto, necessitando ser corrigido para então ser corretamente adicionado no jogo. Ao ser feita a leitura do arquivo de texto sequencialmente, os códigos dos objetos são comparados e então construídos. Para se adicionar um código no projeto basta apenas copiar ele e em seguida, clicar em importar para ele ser validado e então caso tenha sucesso na validação ele será automaticamente adicionado ao projeto. Alguns códigos válidos para testes:

1. AAAAAAAAAAAAAAAAAAAAAAAAAACCFABCCCAAAAACAAAAAADCCC
HCCAGCCAAAAACAAAAACAAAAACCICECICQB0107;
2. BAAAFCCCLAAAAACACCCAAACCAAAACACAAAAACCCCAKALEEEEC
AAAACACAAAAACAACCCACICCCICCCCCCAAQA0107;
3. AAAAAAAAAAAAAAAAAAACCHCCCAAAAAAACAAAFAMAAABCCAG
AAAAACAAAAAACAAAAACAAAAAACCCCCCAAQA0107;
4. AAAAAAAEEAEAAAEEEEEEEEAEAEAEAEBAEEEEEEAEAEAAAAAE
AAAAEAEEEEEEEAEFEEEEAAAAAAAAAAAAAOB0701;

5. AAAAAADCCCCAAAICCDAAACAAACAAACAACAAADEEBCAACEECA
AACAAACAACAAAAAACAAACFADCICICICAAAAOA0913;
6. AAAAAAAAAAAAAAAAAAAAHCCICCAAAAAAAAAACAAAAACAAABCCRC
RGAAlIAAAAAACCAAAAAACAAAAAAAAAAAFRCRAAMA0107;
7. AAACCCAAAAAAAAAAACAAACAAAFAAAAACAAACAAACAABAAACAA
CAACAAAACACAACAACAAAAACAAAACCAAAAAY@01070004192383;
8. AAAAAADGAAAAAAAAADDRDDAAAAAAAAADAAAADASAAAAABAIC
HAEAAAAAAAAACAAAEEAFAAAAAJLCCRCUUAAT@01130000131071;
9. AAAAAANAAAADAAAAAACCCAAATANNAAAACAAAACADAAAABCCRNC
COAAAFAAAAVAAAAACACAAAANVAAAAACAAAV@10140000129151;
10. JEEEEEEEEACaaAEAAAAAAAAEACaaAIEEE_bFAEACAMAAAAAAAEEAC
AAAAABEEEEEIACC`AAAAAAAAAAAAAAAAAO@21120470024188.

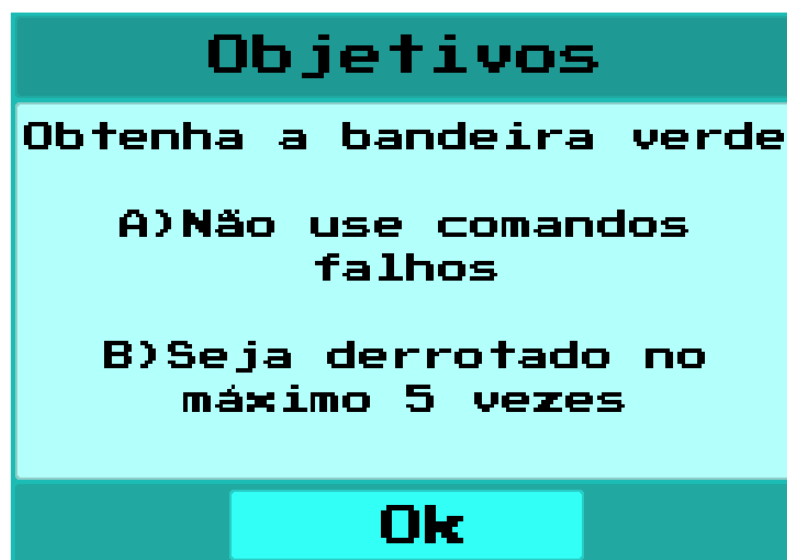
4.2.3 Objetivos e Placar

Outra mecânica interessante adicionada ao projeto é o sistema de pontuação, onde cada nível concluído possui um placar variando de 1 a 3, onde 1 significa que o nível foi apenas concluído, 2 significa que pelo menos um objetivo extra foi concluído e 3, resultando em um nível 100% concluído.

Cada nível possui 3 objetivos, sendo eles capturar a bandeira verde e mais dois outros opcionais, eles variam conforme o nível escolhido, podendo ser observado apenas quando o jogo começa. Os objetivos variam bastante entre si, são exemplos de objetivos disponíveis:

- **Otimização de Algoritmo:** O jogador deve completar o desafio atual, utilizando no máximo **X** comandos;
- **Economia de RAM:** O jogador deve completar o desafio atual, utilizando um máximo de **X** de RAM;
- **Execução Correta:** O jogador pode utilizar até **X** comandos falhos;
- **Programação Cautelosa:** O jogador pode ser derrotado até **X** vezes naquele desafio;
- **Raciocínio rápido:** O jogador deve concluir o desafio em no máximo **X** minutos.

Figura 24 — Exemplo de objetivos de um nível.



Fonte: O Autor (2021).

Os objetivos foram pensados para além de ser uma forma de entretenimento, ser também uma maneira do jogador evoluir suas habilidades, praticando cada vez mais como concluir um desafio proposto da melhor forma mesmo com algumas dificuldades variantes a cada nível concluído. Feitos com o intuito de realmente serem uma maneira de ensinar boas práticas de programação sem que o jogador realmente se dê conta disso, como por exemplo, a otimização de instruções reduzindo o tamanho de código necessário para se fazer o mesmo objetivo, a economia de memória do computador diminuindo gastos desnecessários ou até o desenvolvimento do pensamento rápido para se solucionar um determinado problema.

4.3 Conceitos utilizados no projeto

Foram introduzidos no sistema alguns conceitos de tecnologia de forma a se misturar com o ambiente do jogo, deixando que o seu funcionamento seja apresentado ao jogador sem que ele se dê conta de que foram baseados nos conceitos reais. Foram incluídos no jogo a memória RAM e o HD, por exemplo, como mecânicas funcionais e necessárias para avançar nos níveis.

4.3.1 A Memória RAM

A memória RAM ou memória de acesso aleatório, é uma memória volátil, ou seja, seus novos conteúdos são perdidos quando a energia é interrompida. A RAM possui um funcionamento não sequencial, por isso a origem do seu nome. Sua capacidade é relativamente baixa variando na maioria das vezes entre 2 GB até 16 GB por memória, mas não se limitando a isso. Uma das suas principais vantagens é a velocidade que é extremamente rápida tornando a leitura dos arquivos um trabalho mais eficiente de ser executado.

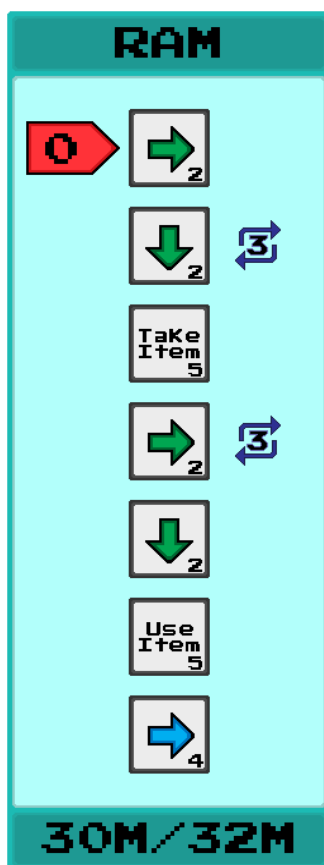
Figura 25 — Foto de memória RAM.



Fonte: Techtudo (2012) <https://www.techtudo.com.br/noticias/2014/09/como-escolher-uma-bom-memoria-ram.ghml>

No projeto, a memória RAM é apresentada como o armazenamento de ações do nosso robô, onde vão ser guardadas as instruções que ele deve executar para conseguir concluir os quebra-cabeças que serão apresentados durante os níveis. Da mesma forma que a memória real de um computador, a memória RAM do CodBot possui baixa capacidade, cada instrução colocada nela consome uma quantidade X de memória, fazendo com que o excesso de comandos trave o robô e ele não consiga executar nenhuma operação até que mais espaço de memória seja liberado.

Figura 26 — Memória RAM representada dentro do projeto.

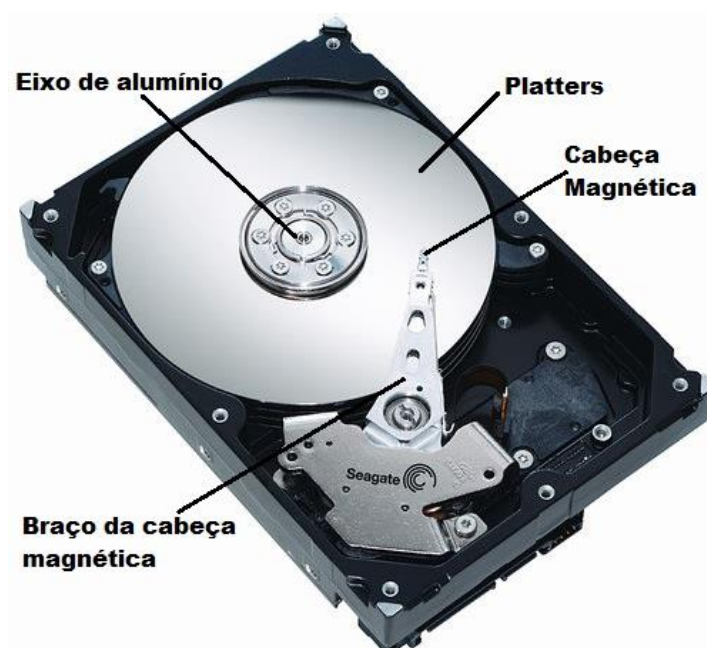


Fonte: O Autor (2021).

4.3.2 O HD

Em contrapartida com a memória RAM, o HD (Hard Disk) ou disco rígido, é uma memória não-volátil que quando tem o seu funcionamento interrompido mantém seus dados armazenados para futuras utilizações. Sua velocidade é extremamente mais lenta que a memória RAM, mas o seu foco não está na velocidade e sim na capacidade de armazenamento que é extremamente alta, variando entre 250 GB e indo até capacidades muito altas como 10 TB ou mais. O HD e a memória RAM trabalham juntos para se conseguir uma velocidade e um armazenamento ideal, a memória RAM servindo como um acelerador de velocidade no computador e o HD fornecendo um grande espaço de armazenamento.

Figura 27 — Foto de um disco rígido.

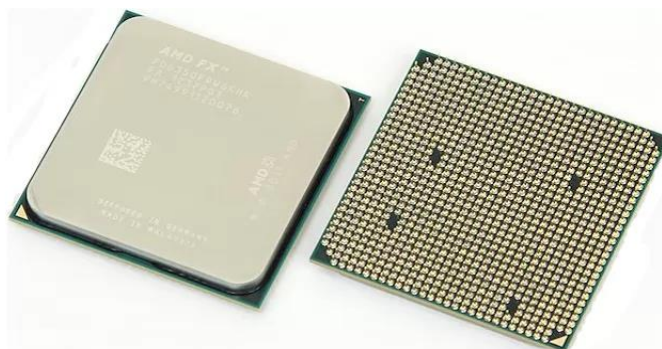


Fonte: Techtudo (2012) <https://www.techtudo.com.br/artigos/noticia/2012/03/para-que-serve-o-hd.html>

4.3.3 A CPU

Uma Unidade Central de Processamento, também chamada de Processador, é um dos componentes mais importantes do hardware, conhecido também como “cérebro do computador”, sendo ele responsável por executar as diversas tarefas da máquina. O CPU possui uma velocidade denominada de Clock, Relógio ou Frequência, que dita a velocidade que as informações vão ser transmitidas pelo sistema, sendo ela medida em Hertz, que estabelece a quantidade de instruções que vão ser executadas pelo processador a cada segundo decorrido. Ou seja, um processador com 3 GHz de velocidade consegue executar até um total de 3 bilhões de instruções em um único segundo. Outra informação extremamente relevante sobre as CPUs, é a quantidade de *Cores* ou Núcleos que ele possui, essa quantidade influencia na quantidade de operações simultâneas que podem ser executadas por aquele processador, começando o processamento de instruções de forma paralela, facilitando a execução de múltiplas tarefas.

Figura 28 — Imagem de um processador AMD.



Fonte: Techtudo (2015) <https://www.techtudo.com.br/noticias/noticia/2015/04/o-que-e-CPU-entenda-qual-e-o-significado-da-sigla-e-sua-importancia.html>

No CodBot, o processador é representado pela quantidade de comandos que o robô pode executar em determinado nível, sendo que, quanto mais avançado o processador utilizado no nível, mais instruções complexas o robô pode executar, aumentando assim a diversidade de funções e dando mais complexidade aos quebra-cabeças que vão sendo apresentados ao longo do avançar do jogo. O *gameplay* do jogo, possui ações de movimento, coleta e uso de objetos, interação com o cenário, entre outras.

Figura 29 — CPU representado dentro do projeto.



Fonte: O Autor (2021)

Alguns outros conceitos também foram utilizados para dar uma estética mais tecnológica e transformar a parte visual do projeto num ambiente mais relacionado a programação, como por exemplo a utilização visual de pastas como opções de menu, os arquivos simbolizando os níveis a serem completados pelo jogador e o título do projeto sendo envolvido pelas tags “</>” utilizadas em HTML (Hypertext Markup Language), uma linguagem voltada para web. Tudo isso feito para ambientar o usuário da melhor forma possível nas áreas tecnológicas com foco no desenvolvimento e na programação.

5 CONCLUSÃO

Tendo em vista os aspectos observados, foi exposta uma maneira interessante e eficaz de introduzir novos conhecimentos através de um formato que vem sendo cada vez mais utilizado atualmente que é a gamificação. Adicionalmente, e complementando o trabalho foi apresentada uma ferramenta desenvolvida com esse propósito, chamada de CodBot, capaz de auxiliar no desenvolvimento e na formulação de soluções utilizando-se de quebra-cabeças.

Espera-se que essa ferramenta estimule a criatividade através dos desafios propostos e criados pelo professor, para seus alunos resolverem da melhor forma que conseguirem, desenvolvendo o raciocínio lógico e possibilitando o exercício do processo de organização mental na medida em que são forçados a pensar de forma semelhante ao funcionamento de um algoritmo. Nesse processo os elementos essenciais estão presentes e as ações primitivas (comandos) são executadas de forma a desenvolver o pensamento lógico considerando as premissas encapsuladas no conceito de algoritmo: sequência, ordenada, finita e não ambígua de etapas que realizam uma tarefa ou resolvem determinado problema. Dessa maneira, pode-se, além de desenvolver o pensamento e a exercitar a organização mental (aprender a pensar!), aumentar a interação (e a comunicação) entre aluno e professor através de desafios regulares.

PROJETOS FUTUROS

Pretende-se continuar desenvolvendo e pesquisando aplicações que ajudem no aprendizado de algoritmos, de programação e de tecnologia em geral. Existe uma demanda grande para projetos nesse segmento e um exemplo desse tipo de aplicação é o *AutomaTaker* para Android que possibilita a representação de autômatos das mais diversas formas no dispositivo móvel. Outra possibilidade são projetos para desenvolvimento de programas que demonstrem visualmente (com qualidade) o funcionamento de certos tipos de algoritmos, como por exemplo, ordenação por bolha, inserção e seleção, estruturas de dados abstratas como filas e pilhas entre outros. Seria desejável e muito útil do ponto de vista acadêmico uma aplicação que implementasse uma máquina de Turing simples para Android e que exibisse visualmente como os comandos aplicados a ela são executados em tempo real, sendo possível visualizar o estado atual da fita, tanto como as suas transições de estados e de direções. Sem dúvida seria um grande aliado para que o aprendizado desse tema seja visualmente intuitivo o que torna seu entendimento uma atividade mais simples e prazerosa.

REFERÊNCIAS

AOSANI, Tânia. TÂNIA AOSANI. O que a Geração Z pode nos ensinar sobre Propósito. [S.l.]. nov. 2020. Disponível em: <https://taniaosanipsicologia.com.br/blog/o-que-a-geracao-z-pode-nos-ensinar-sobre-proposito/>. Acesso em: 13 setembro 2021.

BALDISSERA, Olívia. PÓS PUCPR DIGITAL. O que é gamificação e como ela aumenta o engajamento. [S.l.]. abr. 2021. Disponível em: <https://posdigital.pucpr.br/blog/gamificacao-engajamento>. Acesso em: 12 setembro 2021.

CAIUSCA, Alana. EDUCA MAIS BRASIL. Jogos. [S.l.]. jan. 2019. Disponível em: <https://www.educamaisbrasil.com.br/enem/educacao-fisica/jogos>. Acesso em: 12 setembro 2021.

CHESS. Os 10 Principais Benefícios do Xadrez. [S.l.]. ago. 2018. Disponível em: <https://www.chess.com/pt/article/view/os-10-principais-beneficios-do-xadrez>. Acesso em: 13 setembro 2021.

DAL, Luiz. SUPERINTERESSANTE. O jogo real de UR. [S.l.]. nov. 1990. Disponível em: <https://super.abril.com.br/comportamento/o-jogo-real-de-ur/>. Acesso em: 13 setembro 2021.

DIÁRIO DO COMÉRCIO. Geração Z está além de estereótipos. [S.l.]. nov. 2019. Disponível em: <https://diariodocomercio.com.br/gestao/geracao-z-esta-alem-de-estereotipos/>. Acesso em: 13 setembro 2021.

EDUCAÇÃO INTEGRAL. O jogo real de UR. [S.l.]. jun. 2018. Disponível em: <https://educacaointegral.org.br/glossario/infancia/>. Acesso em: 13 setembro 2021.

EDUCLUB. Mancala – o que é, como se joga e objetivos. [S.l.]. [20-] Disponível em: <https://www.educlub.com.br/mancala-o-que-e-como-se-joga-e-objetivos/>. Acesso em: 12 setembro 2021.

KENZIE ACADEMY. Linguagem de programação: o que é e qual linguagem aprender. [S.l.]. [20-]. Disponível em: <https://kenzie.com.br/blog/linguagem-de-programacao/>. Acesso em: 14 outubro 2021.

LUDOPEDIA. Sobrevivendo aos anos - o jogo real de UR. [S.l.]. set. 2017. Disponível em: <https://www.ludopedia.com.br/topico/16690/sobrevivendo-aos-anos-o-jogo-real-de-ur>. Acesso em: 12 setembro 2021.

LUDOS PRO. Gamificação: o que é e quais os benefícios na aprendizagem? [S.l.]. ago. [20-]. Disponível em: <https://www.ludospro.com.br/blog/o-que-e-gamificacao>. Acesso em: 10 setembro 2021.

NEHAB, Carlos. Apostila Construção de Algoritmos. 2016. p 1-4.

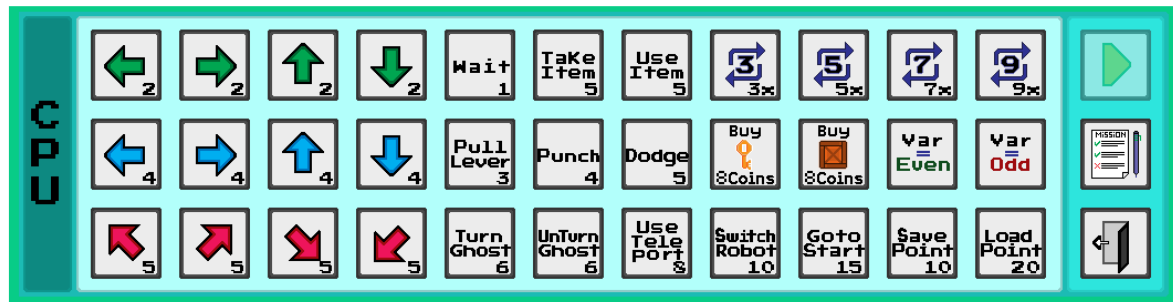
PORFÍRIO, Francisco. BRASIL ESCOLA. Geração Z. [S.l.]. [20-]. Disponível em: <https://brasilescola.uol.com.br/sociologia/geracao-z.htm>. Acesso em: 13 setembro 2021.

RONDINELLI, Paula. BRASIL ESCOLA. Xadrez. [S.l.]. [20-]. Disponível em: <https://brasilescola.uol.com.br/educacao-fisica/xadrez.htm>. Acesso em: 13 setembro 2021.

TÚLIO, Daniel. GAMEGESIS. Senet: o jogo de tabuleiro mais antigo já registrado. [S.l.]. mar. 2021. Disponível em: <https://www.gamegesis.com/2021/03/12/senet-o-jogo-de-tabuleiro-mais-antigo-ja-registrado/>. Acesso em: 12 setembro 2021.

vini. I DO CODE. O que é Programação e qual a sua importância para o futuro digital? [S.l.]. nov. 2016. Disponível em: <https://idocode.com.br/blog/programacao/o-que-e-programacao/>. Acesso em: 16 setembro 2021.

ANEXO A – MANUAL DE COMANDOS



Fonte: O Autor (2021).

Documentação criada para explicar o funcionamento e a utilização de todos os mais diversos comandos disponíveis no projeto CodBot, comentar sobre seus custos, eventuais vantagens e desvantagens de utilização.

Legenda:

Custo: Quantidade de memória RAM em MB necessária se utilizar o comando.

Erro: Execução Inválida de um comando que o torna inútil e ineficaz.

Desafio: Mostra em qual nível básico o comando é explicado.

Comandos Básicos de Movimento. (Ver Níveis 1, 2 e 3)



Esses são os comandos utilizados para mover o robô Cody para todas as 4 direções normais, Esquerda, Direita, Cima e Baixo.

Quando um desses comandos de movimento forem utilizados para se mover para uma direção indisponível, um **ERRO** será gerado.

São os comandos mais básicos que ele pode executar e serão também os mais utilizados para resolver os seus desafios.

Existe também o comando de ficar parado, o robô não executa nada durante apenas um momento, ficando imóvel até a execução do próximo comando.

O custo de RAM desses comandos é bem baixo pois são comandos de alta utilização, **2 MB** para os de movimento e **1 MB** para o de ficar parado e esperar.

Fonte: O Autor
(2021).

Comandos (Pegar / Usar) item. (Ver Níveis 4 e 5)



Objetos cobertos por uma aura branca são classificados como Itens e podem ser utilizados ao seu favor.

Para obter um item, o comando Take Item deve ser utilizado quando o robô Cody estiver no mesmo espaço que o item para conseguir pegar ele e armazenar no HD. Ao pegar dois itens, o item anterior é apagado e não poderá ser mais utilizado.

Fonte: O Autor
(2021).

Para se utilizar um item basta utilizar o comando Use Item quando um item estiver alocado dentro do HD, os seguintes objetos são considerados itens:



Fonte: O Autor (2021).

O custo desses comandos é um pouco mais elevado, mas isso não é um problema porque eles não são tão utilizados assim no mesmo desafio, custando **5 MB** cada.

Caso você tente pegar um item fora do alcance ou utilizar um item com o HD vazio, um **ERRO** será gerado.

Comandos de Repetição.

(Ver Nível 6)



Fonte: O Autor (2021).

Esses comandos não funcionam isoladamente, sendo necessário serem utilizados em conjunto com qualquer outro comando.

Servem como uma estrutura de repetição para repetir um comando anterior N vezes, podendo ser 3, 5, 7 ou 9 vezes repetido.

São utilizados como um mecanismo para acelerar a programação e a utilização de comandos que serão repetidos muitas vezes, basta apenas arrastar uma repetição para um comando já na memória RAM para repetir ele.

Seu custo de RAM varia (3, 5, 7 ou 9) conforme o custo do comando repetido, por exemplo, se um comando que custa 2 MB de RAM for repetido 5 vezes, então o seu custo será de 10MB.

Comandos de Pulo.

(Ver Nível 7)



Fonte: O Autor (2021).

Estes comandos fazem o robô Cody pular sobre um bloco específico para uma das quatro direções normais, Esquerda, Direita, Cima e Baixo.

Só podem ser utilizados no bloco específico escrito “Jump”, caso contrário a execução deste comando vai gerar um ERRO.

Também será gerado um ERRO caso o bloco depois do “Jump” estiver inacessível por algum motivo, ou seja, o segundo bloco na direção escolhida deve estar disponível para se pular.

Seu custo de memória RAM é de 4 MB e são equivalentes a dois movimentos normais para uma mesma direção.

Comando de Acionar Alavanca.

(Ver Nível 8)



Fonte: O Autor (2021).

Serve para (Ativar/Desativar) uma alavanca que está perto do Cody, alavancas (Ativam/Desativam) blocos de mesma cor para abrir caminhos novos, existem alavancas verdes e vermelhas.

Esse comando custa apenas 3 MB de RAM, e causará um ERRO caso não tenha uma alavanca disponível para ser utilizada.

Comando de Socar e Desviar.

(Ver Níveis 9 e 10)



Fonte: O Autor (2021).

O comando de Socar serve para quebrar coisas, como por exemplo caixas ou até alvos azuis e vermelhos.

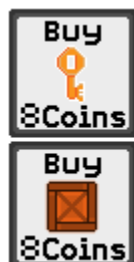
Ao se quebrar certos tipos de caixas, objetos podem ser encontrados nelas como por exemplo chaves e até moedas.

O comando de Desviar deixa o robô Cody intangível por um tempo, fazendo com que ele consiga desviar de projéteis e de danos diversos.

Seus custos em memória RAM são **4 MB** e **5 MB**.

Comandos de Comprar (Chave/Caixa).

(Ver Nível 11)



Fonte: O Autor (2021).

Estes comandos servem para se obter um dos seguintes dois itens que vão diretamente para o seu HD, uma chave que pode ser utilizada para abrir portas ou uma caixa que pode ser utilizada como peso em botões para (ativar/Desativar) blocos.

Em certos desafios moedas podem estar pelo caminho, e é isso que esses comandos usam para serem utilizados.

Os dois comandos utilizam **8 Moedas** como preço de compra em vez de se utilizar RAM, e caso **8 Moedas** não tenham sido corretamente coletadas, um **ERRO** será gerado.

Comandos de Variável (Par/Ímpar).

(Ver Nível 12)



Fonte: O Autor (2021).

São os comandos de funcionamento mais complexo de todo o sistema e só funcionam quando são utilizados em conjunto com outros blocos “Var Pick”, “!Var”, “Even” e “Odd”, não possuem custo de RAM.

Adicionam uma complexidade maior aos desafios utilizando-se de uma variável aleatória que assume o valor de Par ou de Ímpar e só podem ser utilizados junto a outro comando.

Para utilizar esses comandos, o robô Cody deverá pisar em um bloco “Var Pick” para então a variável ser sorteada entre Par ou Ímpar.

Os blocos Even ficam ativos quando a variável está no Par, já os blocos Odd ficam ativos quando a variável está no Ímpar, como não é possível saber qual o valor será sorteado na variável, os dois caminhos devem ser programados, sendo o caminho Even utilizando o comando Even e o caminho Odd utilizando o comando Odd.

Comandos Avançados de Movimento.

(Ver Nível 13)



São comandos de movimento porém diferente dos comandos básicos, servem para se mover diagonalmente entre as plataformas.

Podem ser utilizados para se ultrapassar diagonalmente plataformas que com movimentos normais não são alcançáveis.

Equivalem a dois comandos de movimentos mais um custo extra pequeno por economizarem uma unidade de tempo.

O custo de todos os comandos de movimento diagonal é de **5 MB** de RAM.

Caso o caminho diagonal da direção escolhida não esteja acessível pelo robô, um **ERRO** será gerado.

Fonte: O Autor (2021).

Comandos de (Tornar/Reverter) Fantasma.

(Ver Nível 14)



Devem ser utilizados apenas quando o robô Cody estiver em contato com uma plataforma escrita “Ghost”.

Ao ser utilizado, o robô se (Tornar/Reverter) em um fantasma flutuante que é intangível, ambos os comandos custam **6 MB** de RAM para serem utilizados.

Nessa forma, Cody não pode ser derrotado por objetos e flutua livremente pelas plataformas, podendo ir para qualquer direção sem se importar com os obstáculos em seu caminho, alguns comandos ficam inutilizáveis quando se está em estado de fantasma.

Fonte: O Autor (2021).

Um **ERRO** será gerado caso Cody não esteja em contato com uma plataforma escrita “Ghost”, ou caso tente se reverter sem estar transformado, ou se transformar já estando em estado de fantasma.

Comando de Usar Teletransporte.

(Ver Nível 15)



Esse comando só pode ser utilizado quando o robô Cody está em cima de um Teletransportador (Azul/Vermelho).

Teletransporta o robô diretamente de um Teletransportador para o outro da mesma cor e gerará um **ERRO** caso não exista um.

Fonte: O Autor (2021).

Possui um custo de RAM considerado alto de **8 MB** por ser um comando de movimentação avançado e possibilitar a criação de diversos desafios, locomovendo rapidamente o jogador para diversos lugares.

Comando de Troca de Robô.

(Ver Nível 16)



Fonte: O Autor (2021).

Em alguns desafios um segundo robô fica disponível para ser utilizado na resolução dos quebra-cabeças propostos.

Quando esse comando é utilizado, o controle de um robô passa para o segundo robô e vice-versa.

O segundo robô pode ser utilizado para servir de peso para botões ou resolver outros desafios que o primeiro robô não alcança e vice-versa, sendo necessário o trabalho em conjunto com os dois robôs, seu custo é de 10 MB e gerará um ERRO caso exista apenas 1 robô disponível.

Comando de Voltar ao Início.

(Ver Nível 17)



Fonte: O Autor (2021).

Possui um funcionamento bem simples, quando utilizado teletransporta o jogador para a sua posição inicial, ou seja a posição em que ele começou o desafio atual.

Seu custo de RAM é elevado, custando 15 MB mas dificilmente será utilizado mais de 2 vezes em um mesmo desafio.

Comando de (Salvar/Carregar) Posição.

(Ver Nível 18)



Fonte: O Autor (2021).

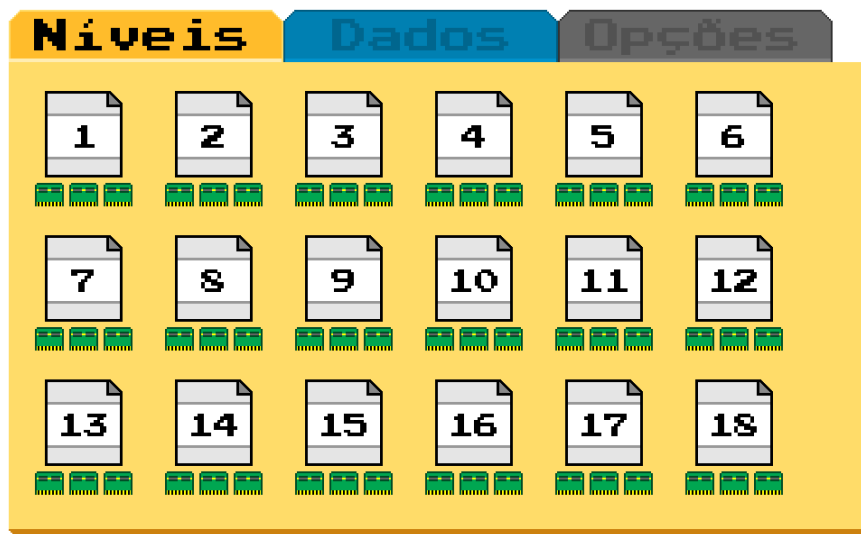
São os últimos comandos a ficarem disponíveis para serem utilizados e igual aos anteriores são relacionados ao teletransporte.

Devem ser utilizados em conjunto e nessa mesma ordem para funcionarem corretamente, primeiramente deve-se salvar a posição atual do robô, para em seguida poder utilizar o comando de carregar a posição antiga do robô.

Um ERRO será gerado caso uma posição inexistente tente ser carregada pelo comando de Load.

Custam respectivamente 10 MB e 20 MB de RAM, sendo esse último o comando com maior custo de RAM de todos os comandos disponíveis.

ANEXO B – MANUAL DE DESAFIOS



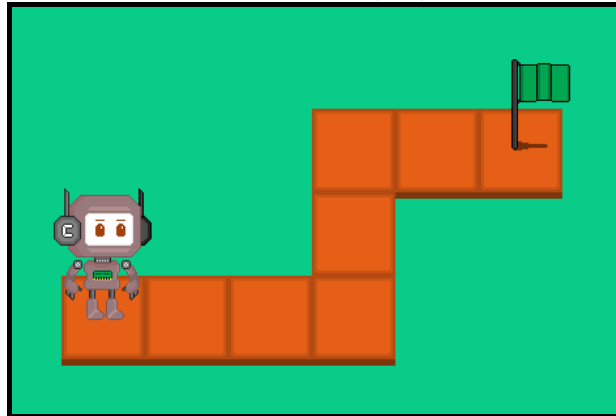
Fonte: O Autor (2021).

Documentação criada para exemplificar como todos os desafios de CodBot podem ser vencidos e como obter a pontuação máxima em cada um deles.

Código dos Comandos:

- 1 - Mover Esquerda
- 2 - Mover Direita
- 3 - Mover Cima
- 4 - Mover Baixo
- 5 - Esperar
- 6 - Pegar Item
- 7 - Usar Item
- (x3) - Repetir x3
- (x5) - Repetir x5
- (x7) - Repetir x7
- (x9) - Repetir x9
- 12 - Pular Esquerda
- 13 - Pular Direita
- 14 - Pular Cima
- 15 - Pular Baixo
- 16 - Puxar Alavanca
- 17 - Socar
- 18 - Desviar
- 19 - Comprar Chave
- 20 - Comprar Caixa
- (E) - Checar Par / EVEN
- (O) - Checar Ímpar / ODD
- 23 - Diagonal Esquerda-Cima
- 24 - Diagonal Direita-Cima
- 25 - Diagonal Direita-Baixo
- 26 - Diagonal Esquerda-Baixo
- 27 - Tornar Fantasma
- 28 - Reverter Fantasma
- 29 - Usar Teletransporte
- 30 - Trocar Robô
- 31 - Ir para Início
- 32 - Salvar Ponto
- 33 - Carregar Ponto

1. Andando (1/2)



Fonte: O Autor (2021).

RAM: 30MB

CPU: Basic

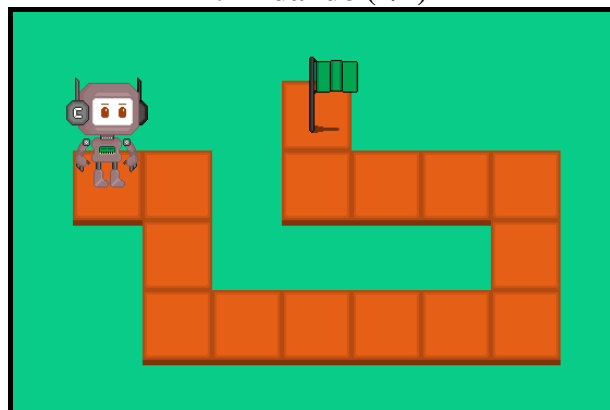
DESCRIÇÃO:

Arraste os comandos de movimento para alcançar a bandeira.

SOLUÇÃO:

2 - 2 - 2 - 3 - 3 - 2 - 2

2. Andando (2/2)



Fonte: O Autor (2021).

RAM: 34MB

CPU: Basic

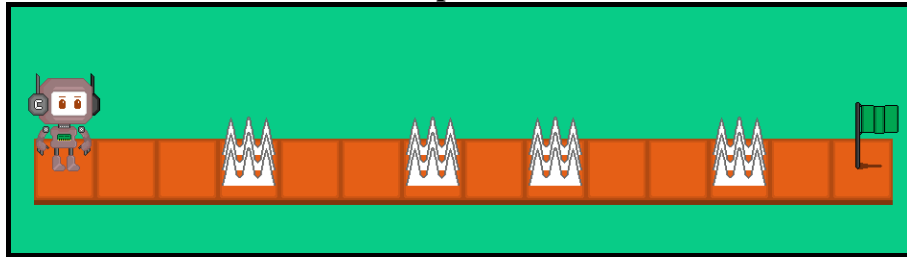
DESCRIÇÃO:

Vamos praticar com mais alguns movimentos simples.

SOLUÇÃO:

2 - 4 - 4 - 2 - 2 - 2 - 2 - 3 - 3 - 1 - 1 - 1 - 3

3. Esperando



Fonte: O Autor (2021).

RAM: 34MB

CPU: Basic

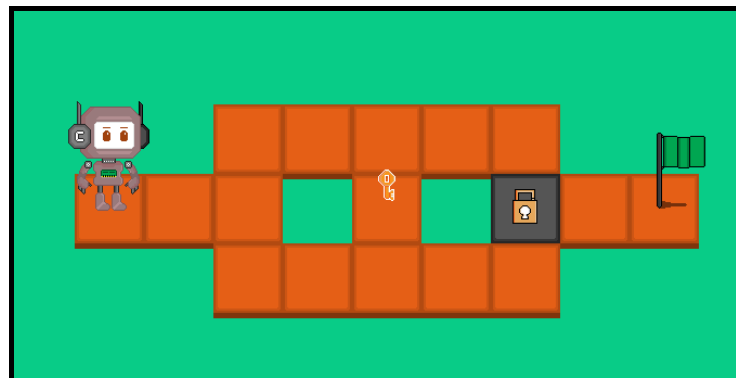
DESCRIÇÃO:

Alguns momentos você só precisa esperar...

SOLUÇÃO:

2 - 2 - 2 - 2 - 2 - 2 - 2 - 2 - 5 - 5 - 2 - 2 - 2 - 5 - 2 - 2 - 2

4. Itens - Chave



Fonte: O Autor (2021).

RAM: 40MB

CPU: Basic+

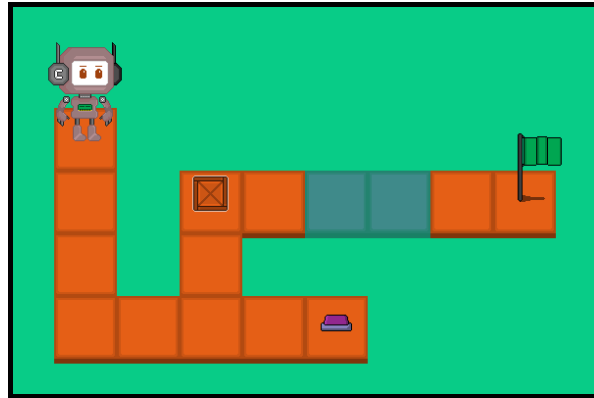
DESCRIÇÃO:

Alguns caminhos só podem ser abertos com chaves, pegue uma e use na fechadura.

SOLUÇÃO:

2 - 2 - 3 - 2 - 2 - 4 - 6 - 3 - 2 - 2 - 7 - 4 - 2 - 2

5. Itens - Caixa



Fonte: O Autor (2021).

RAM: 55MB

CPU: Basic+

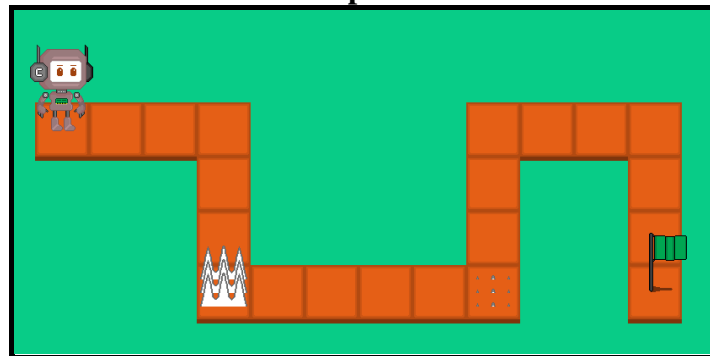
DESCRIÇÃO:

Coloque um peso em cima do botão para ativar mecanismos com ele, uma pequena caixa pode lhe ajudar com isso.

SOLUÇÃO:

4 - 4 - 4 - 2 - 2 - 3 - 3 - 6 - 4 - 4 - 2 - 2 - 7 - 1 - 1 - 3 - 3 - 2 - 2 - 2 - 2 - 2

6. Repetindo



Fonte: O Autor (2021).

RAM: 50MB

CPU: Logic

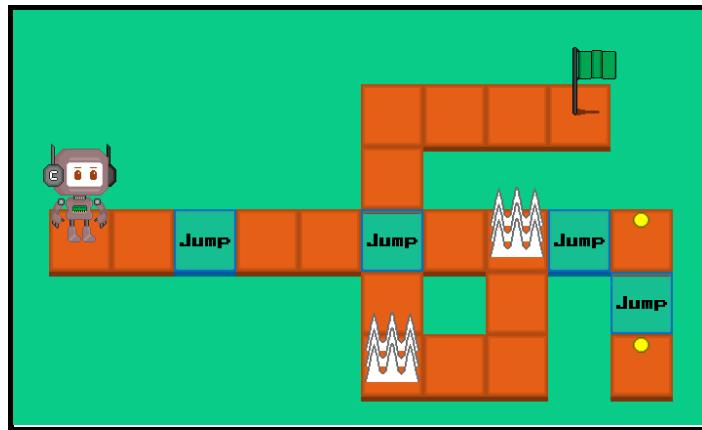
DESCRIÇÃO:

Algumas vezes, repetir a mesma ação pode ser a solução ideal.

SOLUÇÃO:

2(x3) - 4(x3) - 5 - 2(x5) - 3(x3) - 2(x3) - 4(x3)

7. Pulando



Fonte: O Autor (2021).

RAM: 55MB

CPU: Jumper

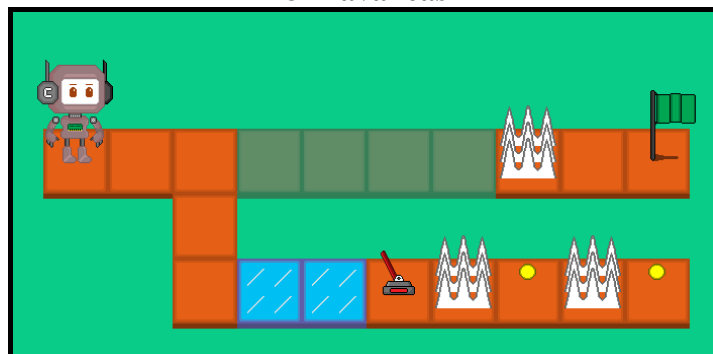
DESCRIÇÃO:

Para passar por certos caminhos, basta apenas pular por cima deles.

SOLUÇÃO:

2 - 13 - 2 - 13 - 5 - 2 - 13 - 15 - 14 - 12 - 4 - 4 - 1 - 1 - 3 - 14 - 3 - 2(x3)

8 Alavancas



Fonte: O Autor (2021).

RAM: 60MB

CPU: RPG

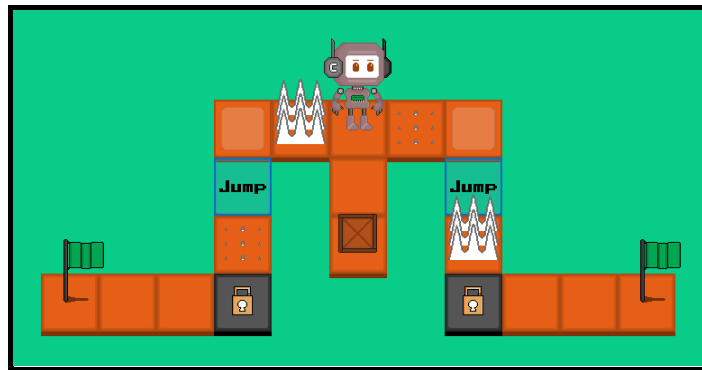
DESCRIÇÃO:

Alavancas podem ativar ou desativar blocos para te ajudarem com o seu caminho.

SOLUÇÃO:

2 - 2 - 4 - 4 - 2 - 16 - 2 - 2 - 5 - 2 - 2 - 5 - 5 - 1 - 1 - 5 - 5 - 1 - 1 - 1 - 3 - 3 - 2(x5) - 2 - 2

9. Punhos



Fonte: O Autor (2021).

RAM: 60MB

CPU: RPG

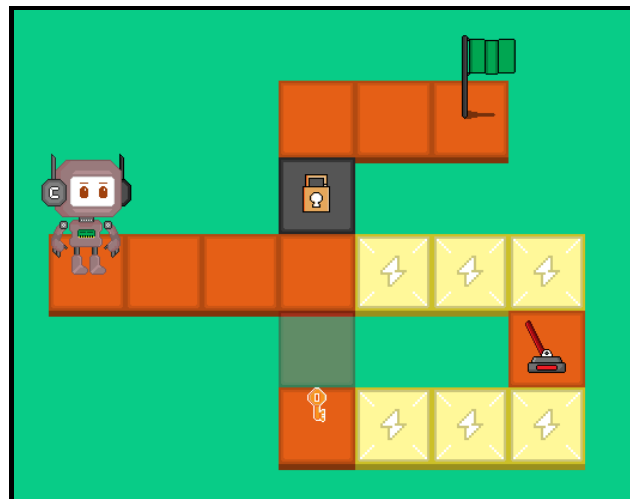
DESCRIÇÃO:

Seus socos podem ser usados para quebrar coisas.

SOLUÇÃO:

4 - 17 - 4 - 6 - 3 - 3 - 1 - 1 - 5(x3) - 15 - 7 - 4 - 1(x3)

10. Desviar



Fonte: O Autor (2021).

RAM: 65MB

CPU: RPG

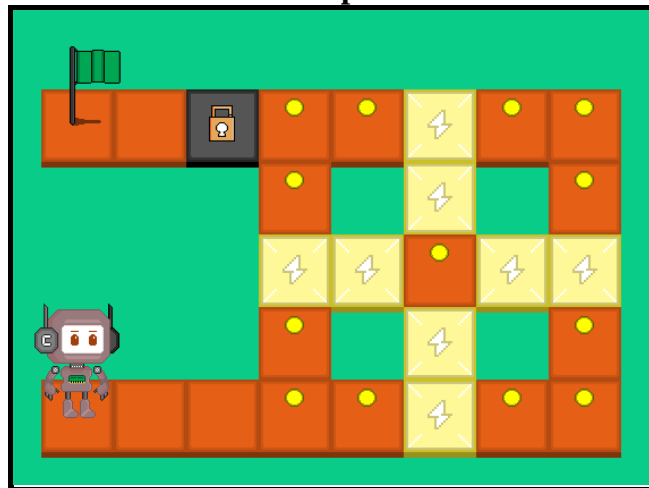
DESCRIÇÃO:

Às vezes a única saída é desviar para se manter em segurança.

SOLUÇÃO:

2(x5) - 18 - 2 - 4 - 16 - 4 - 1 - 18 - 1 - 1 - 6 - 3 - 3 - 7 - 3 - 3 - 2 - 2

11. Comprando



Fonte: O Autor (2021).

RAM: 70MB

CPU: RPG

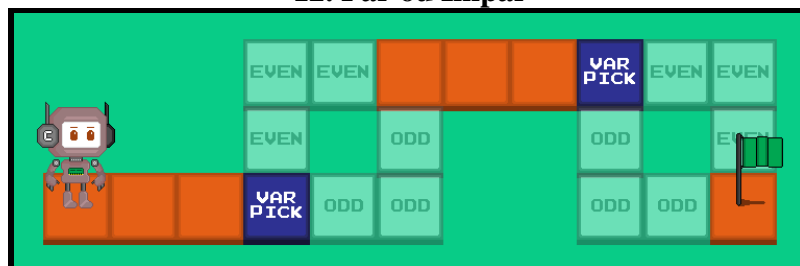
DESCRIÇÃO:

As moedas podem lhe ser úteis para obter chaves ou caixas, basta ter a quantidade certa.

SOLUÇÃO:

2(x3) - 3 - 4 - 2 - 2(x3) - 3 - 3 - 18 - 1 - 1 - 5 - 2 - 2 - 18 - 3 - 3 - 1 - 5 - 1(x3) - 4 - 3 - 19 - 7 - 1(x3)

12. Par ou Ímpar



RAM: 60MB

CPU: RPG+

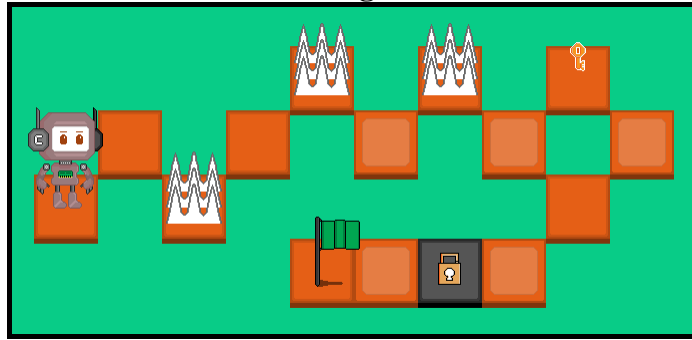
DESCRIÇÃO:

Às vezes o caminho é imprevisível, você precisa programar as duas possibilidades e esperar o resultado.

SOLUÇÃO:

2(x3) - 2(O) - 2(O) - 3(O) - 3(O) - 3(E) - 3(E) - 2(E) - 2(E) - 2(x3) - 2(E) - 2(E) - 4(E) - 4(O) - 4(O) - 4(O) - 2(O) - 2(O)

13. Diagonais



Fonte: O Autor (2021).

RAM: 85MB

CPU: XFast

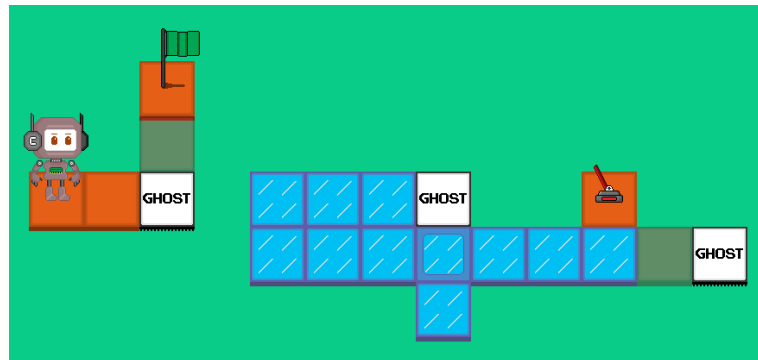
DESCRIÇÃO:

Existem caminhos que só podem ser alcançados com movimentos diagonais.

SOLUÇÃO:

24 - 25 - 24 - 5 - 5 - 24 - 25 - 5 - 5 - 24 - 25 - 24 - 6 - 25 - 26 - 26 - 7 - 1(x3)

14. Fantasma



Fonte: O Autor (2021).

RAM: 90MB

CPU: XFast+

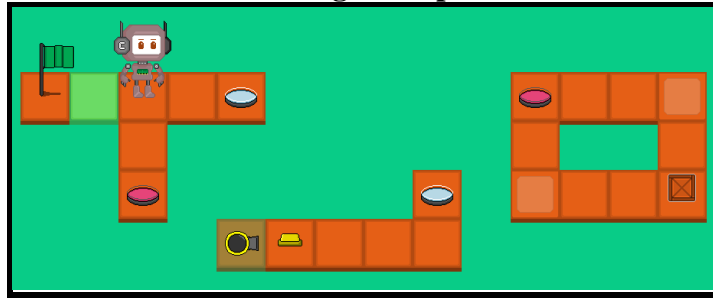
DESCRIÇÃO:

Pra quê se preocupar com plataformas, se você pode sair por aí flutuando por tudo?

SOLUÇÃO:

2 - 2 - 27 - 2(x5) - 28 - 1 - 4 - 2 - 3 - 16 - 4 - 2 - 2 - 27 - 1(x9) - 23 - 28 - 3 - 3

15. Viagem Rápida



Fonte: O Autor (2021).

RAM: 120MB

CPU: Omega

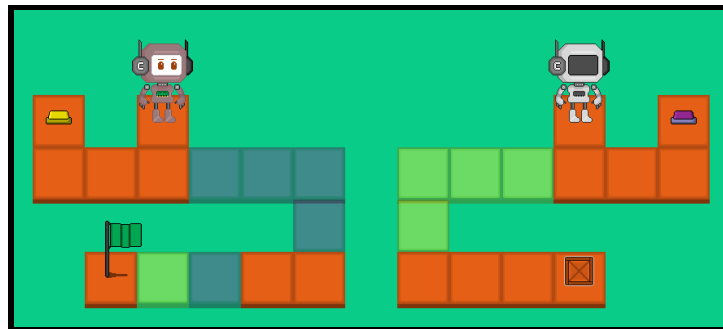
DESCRIÇÃO:

Um Teletransportador podem ser utilizados para alcançar rapidamente certos pontos e retornar por eles.

SOLUÇÃO:

4 - 4 - 29 - 2(x3) - 4 - 4 - 6 - 1(x3) - 3 - 3 - 29 - 3 - 3 - 2 - 2 - 29 - 4 - 1 - 1 - 18 - 1 - 7 - 2 - 18 - 2 - 2 - 3 - 29 - 1(x3) - 1

16. Uma Mãozinha



Fonte: O Autor (2021).

RAM: 120MB

CPU: Omega

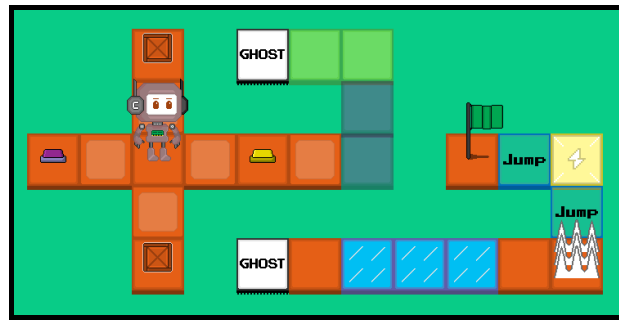
DESCRIÇÃO:

Um ajudante extra pode te ajudar a resolver alguns desafios, experimente utilizar ele.

SOLUÇÃO:

4 - 1 - 1 - 3 - 30 - 4 - 1(x3) - 4 - 4 - 2(x3) - 6 - 1(x3) - 3 - 3 - 2(x5) - 3 - 30 - 7 - 4 - 2(x5) - 4 - 4 - 1(x3) - 1

17. Retornando



Fonte: O Autor (2021).

RAM: 140MB

CPU: Omega

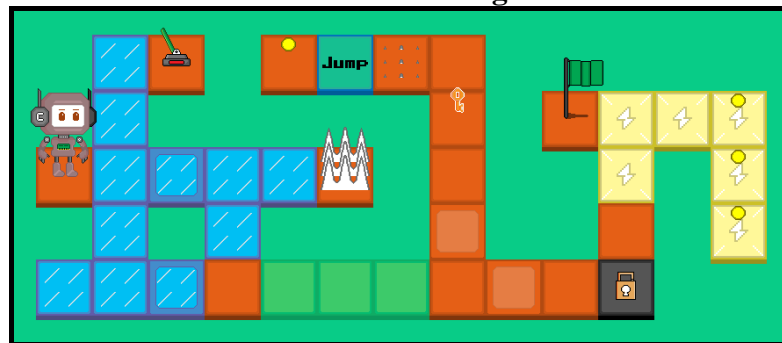
DESCRIÇÃO:

Se não sabe o que fazer em seguida, experimente começar novamente do começo.

SOLUÇÃO:

3 - 3 - 6 - 31 - 1 - 1 - 7 - 31 - 4 - 4 - 6 - 31 - 2 - 2 - 7 - 2 - 2 - 3 - 3 - 1 - 1 - 27 - 4(x3) - 4 - 28 - 2(x3) - 14 - 12

18. Salva/Carrega



Fonte: O Autor (2021).

RAM: 180MB

CPU: Omega

DESCRIÇÃO:

Saiba o momento exato para Salvar um ponto e depois para o Carregar. cuidado onde você está salvando.

SOLUÇÃO:

5 - 2 - 1 - 4 - 32 - 1 - 2 - 3 - 2 - 16 - 33 - 2(x3) - 2 - 32 - 3(x3) - 6 - 3 - 5 - 5 - 1 - 12 - 33 - 2 - 2 - 7 - 2 - 3 - 3 - 18 - 3 - 2 - 18 - 2 - 4 - 18 - 4 - 3 - 18 - 3 - 1 - 18 - 1 - 1